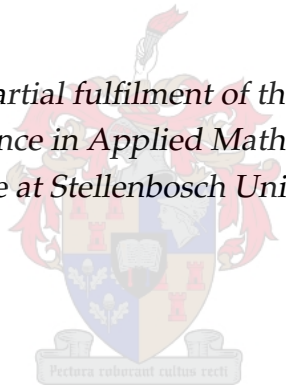
Handover in a Distributed System of UAVs: Application to Wildlife Monitoring

by

Juliana Thomasia Chakirath Marcos

Thesis presented in partial fulfilment of the requirements for the degree of Master of Science in Applied Mathematics in the Faculty of Science at Stellenbosch University



Department of Mathematical Sciences,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Supervisor: Dr Simukai W. Utete

December 2020

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Signature: .....
Juliana Thomasia Chakirath Marcos

Date: December 2, 2020 .....

Abstract

Handover in a Distributed System of UAVs: Application to Wildlife Monitoring

Juliana Thomasia Chakirath Marcos

Department of Mathematical Sciences,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Thesis: MSc. (Applied Mathematics)

December 2020

Wildlife surveillance is of significant interest for the protection of animals and their habitats. In this study, a distributed system of unmanned aerial vehicles (UAVs) or drones is designed for single-animal tracking in terrestrial settings. The system involves four main components which constitute key contributions of the study. The main component is a visual object tracking approach based on the use of a particle filter that switches between measurements from two sources: a simple and fast approach based on colour image segmentation and a slower but more sophisticated method based on a deep learning object detector, the third version of the You Only Look Once detector (YOLOv3). The particle filter switches between the measurement sources using the structural similarity (SSIM) index from the image-processing literature. The SSIM index is also applied in the study for handover of tracking between a pair of drones. Some of the components of the monitoring system have been simulated using wildlife footage recorded by drone (obtained from an animal behaviour group). Extensive simulation tests were carried out during the study. These demonstrate, amongst other results, that better real-time object detection is obtained by replacing YOLOv3 by techniques such as boosting and channel and spatial reliability tracking (CSRT). The design developed and components tested suggest some directions for single-animal tracking by a distributed system of drones.

Keywords: Animal tracking algorithm, boosting, channel and spatial reliability tracking (CSRT), drone, handover, multiple instance learning (MIL), particle filter, structural similarity (SSIM), unmanned aerial vehicle (UAV), You Only Look Once version 3 (YOLOv3).

Opsomming

Oorhandiging in 'n verspreide stelsel van UAV's: Toepassing op wildmonitering

(" Handover in a Distributed System of UAVs: Application to Wildlife Monitoring ")

Juliana Thomasia Chakirath Marcos

Departement Toegepaste Wiskunde,
Universiteit van Stellenbosch,
Privaatsak X1, Matieland 7602, Suid Afrika.

Tesis: MSc. (Wiskunde)

Desember 2020

Die waarneming van wildslewe is van opmerklike belang vir die konservasie van diere en hul habitatte. Hierdie studie maak gebruik van 'n verspreide stelsel wat bestaan uit onbemande lug voertuie (UAVs), of 'drones,' wat ontwerp is om 'n enkele dier te agtervolg in aardse instellings. Die sisteem bestaan uit vier hoof komponente, en hierdie komponente is die sleutel bydraers vir die studie. Die belangrikste komponent is 'n visuele voorwerpopsporings tegnologie gebaseer op deeltjiefilter wat wissel tussen metings vanaf twee bronne: 'n eenvoudige en vinnige benadering wat gebaseer is op kleurprintsegmentering en 'n stadiger, maar meer gesofistikeerde, metode wat gebaseer is op 'n diep detektor vir leervoorwerpe, die derde instelling van die 'You Only Look Once' program (YOLOv3). Die deeltjiefilter skakel tussen die meetbronne deur gebruik te maak van die strukturele ooreenkoms (SSIM) indeks uit die beeldverwerkingsliteratuur. Die SSIM-indeks word ook toegepas in die studie vir die oorhandiging van die opsporings proses tussen hommeltuie. Sommige van die komponente van die moniteringstelsel is gesimuleer met behulp van wild beeldmateriaal wat deur hommeltuie geneem is (verkry van 'n dieregedraggroep). Uitgebreide simulasietoetse is tydens die studie uitgevoer. Dit toon onder andere aan dat beter intydse voorwerpopsporing verkry word as YOLOv3 vervang is met tegnieke soos die versterking en opsporing

van kanaal- en ruimtelike betroubaarheid (CSRT). Die ontwerp wat ontwikkel is, en die getoetste komponente, dui op 'n paar moontlike aanwysings vir die gebruik van verspreide hommeltuig sisteeme om enkeldiere te monitor.

Sleutelterme: Diereopspoor algoritme, Versterking, kanaal- en ruimtelike betroubaarheidsopsporing (CSRT), hommeltuig, oorhandiging, meervoudige instansie leer (MIL), deeltjie filter, Strukturele ooreenkoms (SSIM), onbemande lugvaartuig (UAV), *You Only Look Once version 3* (YOLOv3).

Acknowledgements

I firstly give glory to God for all his grace and help which have allowed me to successfully face the challenges encountered during the development of this project.

I wish to express my profound gratitude and my sincere thanks to my supervisor, Dr Simukai W. Utete for her guidance, her support, and her insightful remarks throughout the period of the research.

Next, I would like to thank the Max Planck Institute of Animal Behavior (MPI-AB) in Germany, in particular, the Department of Prof. Iain Couzin and member Dr Blair Costelloe for providing me with wildlife drone footage[1] under an MOU.

I would also like to thank the Centre of High Performance Computing (CHPC) in South Africa for giving me access to computing resources.

My gratitude goes to my thesis examiners for their constructive comments that improved the final version of my thesis.

My sincere gratitude goes to AIMS-SA for the opportunity that was given to me for doing this project and to all AIMS staff, academic and non-academic, who have contributed in one or another way to my unforgettable stay at AIMS. I thank in particular Prof. Jeff Sanders for the time and the knowledge shared during the *Wednesday Group Meetings*. I would also like to express my gratitude to Igsaan, the facility team, and the kitchen staff for their work which has eased the period of the Lockdown which was set up by the South African government in response to the COVID-19 pandemic.

To all my friends and colleagues of the research centre and especially: Evander, Funmilayo, Karabo, Kiprono, Rojo, Samuel, Shankar, and Thabani, I say thank you for the good moments we have spent together and especially those at the tea/lunch breaks. I particularly thank Kibidi and Grisel for the translation into Afrikaans of the abstract of my thesis. To all my other friends, in particular: Amanda, Aulan, and Romaric, I say thanks for your support.

I cannot forget my uncle Mr Philibert Marcos who has encouraged me in pursuing my studies.

Finally, I would like to thank my dear mother, Mrs Agnès Zonon Marcos, and my siblings, Anita, Grace, Julio for their support and their unconditional love.

[1]Information from dataset providers: the dataset was collected with support from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No. 748549. All procedures for collecting the drone footage dataset were reviewed and approved by Ethikrat, the independent Ethics Council of the Max Planck Society. The dataset was collected with the permission of Kenya's National Commission for Science, Technology and Innovation (NACOSTI/P/17/59088/15489 and NACOSTI/P/18/59088/21567) using drones operated by Blair Costelloe with the permission of the Kenya Civil Aviation Authority (authorization numbers: KCAA/OPS/2117/4 Vol. 2 (80), KCAA/OPS/2117/4 Vol. 2 (81), KCAA/OPS/2117/5 (86) and KCAA/OPS/2117/5 (87); RPAS Operator Certificate numbers: RPA/TP/0005 AND RPA/TP/000-0009).

Dedications

I dedicate this work to my late father Mr José Francis Marcos, to my dear mother Mrs Agnès Zonon Marcos and my dear nephew Noah Badarou.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Background and Problem Statement

Over the last two centuries, several animal species have become endangered, or even in some cases extinct, due to human activity. Some of the extinct species are the West African black rhinoceros, the Tasmanian tiger, the Quagga that is a native South African zebra, the Pyrenean Ibex, and the Javan tiger [1]. The development of reliable systems for wildlife monitoring is important both for conservation and preservation efforts and for the development of a better understanding of the interactions of animals with each other and with their environments.

Over the years, researchers have developed tracking systems such as GPS, radio, and satellite transmitters embedded in collars or other devices that can be placed on animals. However, most of them present drawbacks. Firstly, it can be difficult and dangerous to capture wild animals to fit them with such devices. In addition, the process can also be risky for them as it generally requires anesthesia [2]. Secondly, some devices can lead to changes in animals' behaviour and/or threaten their survival and therefore compromise data collected. In [3], it was noticed that slightly heavier collars compared to reference collars, reduced the traveling rate of plains zebras by 50 % during foraging, with both collars' weights within the accepted norm. The researchers in [4] fitted 269 randomly chosen caribou with either light or heavy collars and they noticed that the heavier collar reduced by 18 % annual survival of caribou already in poor body condition.

Unmanned aerial vehicles (UAVs) or drones can offer an alternative approach which overcomes some of these disadvantages:

- drones represent a good non-invasive alternative for wildlife monitoring and surveying, subject to certain conditions on altitude ($>$ 60m AGL), drone horizontal

distance ($<$ 100m) [5], drone approach angle, colour and speed [6].

- Apart from cameras, other types of sensors such as thermal-imaging cameras can be mounted on drones to collect additional data.
- Drones are cheaper [7] and more manoeuvrable in dense areas than other platforms such as satellites and balloons.

However, drones used in such studies are typically civilian drones with limited computing power and battery life. To compensate for these limitations, it is proposed here to use several drones arranged in formation to perform single-animal tracking. The animals under observation with the present drone formation are primarily taken to be terrestrial animals. The formation of drones involves a fixed and virtual grid that covers the living area of the animals under consideration. Drones are positioned at the boundary between adjacent grid cells. They are grounded at their positions and do not hover or fly. It is assumed that they can be in a *Sleep* mode amongst other modes which are described in Subsection 4.1.3 of Chapter 4. Figure 1.1 illustrates a formation where there are 9 cells and therefore 12 drones. A ground station is included in the system, primarily to handle video storage and tracking initialization. It is also assumed that tracking initialization is made by a human operator. Training strategies for machine learning networks could be leveraged to perform tracking initialization automatically. All assumptions made to facilitate the resolution of single-animal tracking with drones are presented in Subsection 4.1.5 in Chapter 4. The present configuration has been chosen amongst others that were proposed for reasons explained in Subsection 4.1.1 in Chapter 4. Also, drones have other drawbacks (e.g. regulations, battery charging requirement, etc.) which are presented in Section 2.3 in Chapter 2 with potential solutions.

In this way, based on visual information, a target animal could be tracked from one cell to another with drones relaying the tracking to each other when necessary, up to tracking termination. At a time step, only a maximum of two drones is flying while the other drones are grounded at their initial positions where they can be charged. Only one drone can follow the target animal. This drone is referred to as the drone-tracker. In this design, the relay partner of the drone-tracker is selected by the drone-tracker based on information such as the actual cell the target is in, the drones available amongst those responsible for that cell, their distances to the drone-tracker, their energy levels, and their memory capacities. The selected drone is addressed as drone-partner or drone-peer. After its selection, the drone-partner moves to the appropriate position to perform the target relay which is simply referred to as handover in this work.

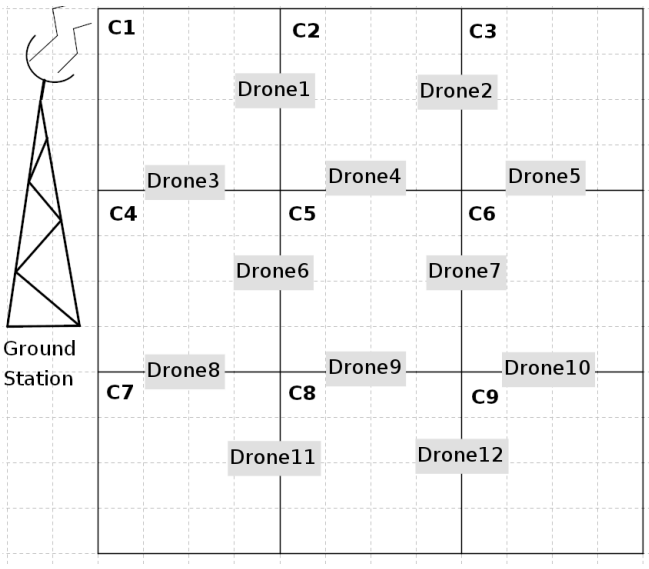


Figure 1.1: Drone formation which involves a drone at the boundary between two adjacent cells. The present configuration illustrates the case where there are 12 drones, in a 3×3 arrangement. The grid has 9 cells labelled C1,C2,...,C9.

It is very important in the proposed monitoring system to estimate the target's position in the images captured by the drone-tracker. Indeed, this is required to direct the drone in its tracking task. The drone-tracker moves in the field during a tracking based on the motion of the target. Simply put, the target's trajectory is the *mirror* of the drone-tracker's trajectory. Moreover, animals' trajectories could be reconstituted offline with successive position estimates to be analysed for insights about the animals' behaviour.

The study is concerned with single-animal tracking as opposed to multi-animal tracking. It is important to emphasize a few points which justify the utility of single-animal tracking in wildlife monitoring.

- Single-animal tracking is relevant for animals that live in groups although some of the members of the group considered might not always be filmed by a drone-tracker, when the drone-tracker is following an individual of the group. For instance, if one is interested in monitoring a group of elephants, the ideal target to track would be the matriarch. On the other hand, if one is interested in the trajectories of the other animals in video captured during single-animal tracking, a given method could be employed for tracking offline.

- For solitary animals or endangered animals with only a few individuals remaining, the monitoring system could be employed to track a single individual. In ad-

dition, the behaviour of a single animal could be representative of the behaviour of the species it belongs to.

- Single-animal tracking is more adapted to the limited on-board capacities of civilian drones. One could extend the system to perform multiple-animal tracking by centralizing computations at the GS, and by employing a latency-free and secured communication system between drones and the GS.

The estimation of the target's location in pixels in video frames captured by the drone-tracker, is typically a visual object tracking (VOT) problem which is one of the most challenging areas of computer vision (CV). VOT solutions generally involve two components: the motion model used to predict the target's location in the next video frame, and the appearance model used to describe the target. Techniques used to model the target's appearance can be grouped into generative and discriminative methods. The former searches the most similar regions [8] while the latter differentiates the foreground from the background [9].

Earlier tracking algorithms use hand-crafted features such as colour and histogram of oriented gradients (HOG) [10] to represent appearance models. However, these methods are not robust enough to handle severe changes in appearance [11] compared to modern deep learning-based trackers [9] such as generic object tracking using regression networks (GOTURN) [12]. Deep learning-based object detectors can be included in modern tracking algorithms since they can be employed with motion models to track objects in successive video frames. Some examples of these detectors are: OverFeat [13], region-based convolutional neural network (R-CNN) [14]. Yet, unlike traditional tracking algorithms, deep learning-based algorithms require significant computation which might be beyond the capacity of on-board processors on some drones.

In this study, it is proposed to take advantage of both paradigms by combining them to track animals using drones. In the proposed VOT solution, a colour segmentation method and the third version of the You Only Look Once (YOLOv3) [15] are switched after a change is detected in the appearance of the target animal based on the structural similarity (SSIM) index [16], in order to provide measurements to the particle filter (PF) algorithm [17] which is used by the drones for tracking. The PF is used to estimate an animal's position in video frames since the target animal is not fitted with tracking devices. The choice of the PF compared to other estimation methods like the Kalman filter is because the PF is not restricted in terms of non-linearity and non-Gaussian distributions [17]. The PF offers more flexibility in changing some components such as the motion model. Additionally, the proposed VOT solution could be extended easily to

multi-animal tracking. Yet, for this study, the Kalman filter can also be used instead of the PF. Note also that during the development of this study the fourth version of YOLO (YOLOv4) was released.

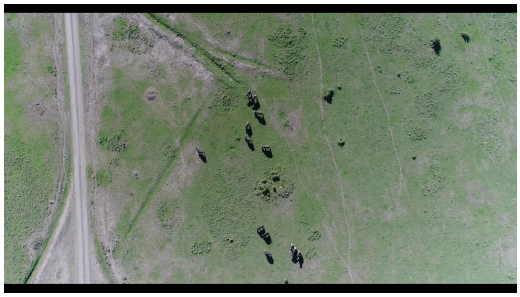
The proposed VOT solution is implemented and tested using primarily a dataset of wildlife footage recorded in Kenya under appropriate permissions by an animal behaviour group. The reader is invited to consult Subsection 5.1.1 for details about the dataset as well as the *Acknowledgements* for information about the permissions and conditions for data collection. Figure 1.2 shows frames extracted from four video sequences of this dataset. They are utilized for testing of the proposed VOT solution and the methods for the validation of the target's identity during handover. Owing to the use of this dataset, challenges that could arise in the field are highlighted. These are for example the small-scale of the animals due to filming from high altitude (to mitigate wildlife disturbance), the impact of filming time which can result in the presence of shadows, and the difficulty for trackers to discriminate animals seen from above. For proof of concept, the proposed VOT solution is tested on a video referred to as the *cows video*[2]. The results obtained are shown in Figure 1.3. The proposed VOT solution also involves a region of interest (ROI) where YOLOv3 looks for the target when called by the SSIM index.

1.2 Objectives of the Project

This project aims to design a system based on a formation of drones for wildlife surveillance and preservation. The system is primarily intended for terrestrial mammals. More specifically, the study aims to:

- design the drone formation,
- address the challenging task of tracking an animal in video frames and in real-time by proposing a framework suitable for drones,
- propose a technique for two drones to relay a tracking to each other,
- propose a potential solution for directing the drone-tracker in the real-world frame using the outputs produced by the VOT framework in video frames.

[2]The *cows video* is available at this address: https://github.com/mpatacchiola/deepgaze/tree/master/examples/ex_particle_filter_object_tracking_video.



(a) Sequence $ob002 - 01a$.

(b) Sequence $ob002 - 01b$.

(c) Sequence $ob003 - 01$.

(d) Sequence $ob090 - 03$.

Figure 1.2: Examples of video frames from the video sequences used in this study for testing the proposed VOT solution and the methods for the verification of the target identity during the handover process. The video sequences are extracted from the videos labelled $ob002 - 01$, $ob003 - 01$ and $ob090 - 03$ of the dataset used in this work. More details about the dataset are available in Subsection 5.1.1.

1.3 Contributions of the Study

The contributions of this study are listed below. Some of these contributions involve the design of algorithms.

- Introduction of the drone formation which is a fixed, virtual grid where drones are positioned at specific locations to ensure that the field of the animals to monitor is well-covered.

- Proposal of a VOT framework which is based on the use of a PF. The latter employs a basic motion model for updating the particles' positions at each time step. On the other hand, the PF can receive measurements from one of two sources. At a time step, the measurement provider is selected based on the value of the SSIM index between the current and the initial appearance of the target. The SSIM index was originally designed to assess the quality of distorted images compared to reference images. Yet, the SSIM index is utilized to compare the structural information be-

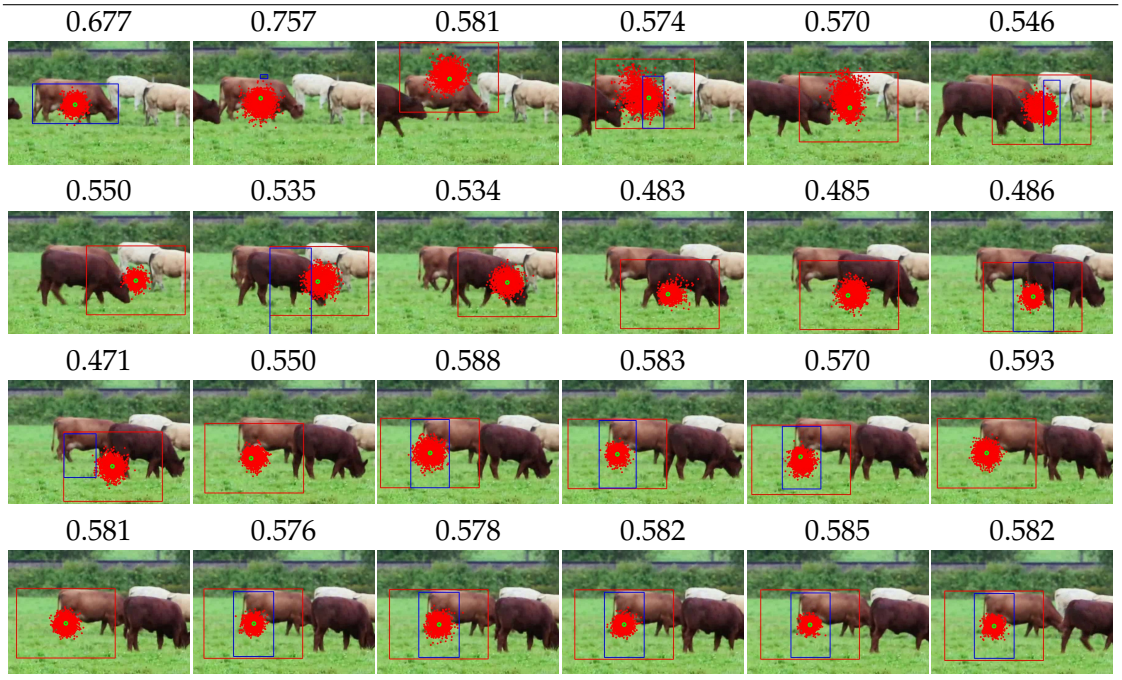


Figure 1.3: Example of occlusion handling with the proposed VOT approach on the *cows video* for a SSIM threshold of 0.6. Particles are represented in red, bounding box in blue, and regions of interest for YOLOv3 in red. Frames are displayed from left to right and from top to bottom starting with the 518^{th} frame with interval 13 frames. Above each frame is the corresponding SSIM value. These values are less than the set threshold in most frames displayed. This justifies the call of YOLOv3 as opposed to the colour image segmentation. YOLOv3 did not detect the cow in some of these frames. The head and the feet of the cows are salient features for YOLOv3.

tween two images, similarly to other studies, in particular, the work of [18] which used the SSIM measure for switching detection and tracking methods. The two measurement providers involved in this study are primarily the colour image segmentation and YOLOv3. However, during the test phase, YOLOv3 is replaced by other trackers to better analyse the performance of the proposed tracking framework. These are essentially machine learning and correlation-filter based trackers.

- Proposition of the concept of relaying a tracking from the drone-tracker to the drone-partner by utilizing the SSIM index to ensure that the target is the same in the field of view (FOVs) of the two drones. This step necessitates the joint-use of the particles from the drone-tracker and the information of the drone-peer's FOV. Prior to handover, an algorithm is proposed for the selection of the most suitable drone-partner amongst drone-candidates responsible for the current cell.

Moreover, other algorithms are designed for the definition of the actions to be taken by the two drones based on the knowledge each drone has about the other in order to ensure the success of handover and to avoid having several drones tracking a target simultaneously.

- Development of a method which uses the outputs of the proposed VOT framework for periodic computation of the yaw and pitch for the drone-tracker's displacement during tracking when it is not hovering above the animal.

1.4 Project Outline

The rest of the thesis is organised as follows: Chapter 2 presents a brief introduction of drones, their categorization and components. Chapter 3 provides a succinct survey of VOT. All of the constituents of the monitoring system are presented in Chapter 4. The main contributions of the study are presented in this chapter. Chapter 5 covers the tests and the results for some of the components of the monitoring system and for other well-known tracking methods. The monitoring system is not implemented as a whole. The components tested or simulated are the proposed VOT solution, the handover communication protocol, and the mechanism for the validation of the identity of the target. Chapter 5 also presents a discussion of the results obtained. Finally, Chapter 6 concludes the study and opens up future perspectives for this work.

Chapter 2

Unmanned Aerial Vehicles

This chapter reviews unmanned aerial vehicles (UAVs) or drones since they are at the core of the proposed tracking system. Indeed, in order to develop a monitoring system that could be implemented in real life, it is required to make design decisions that are suitable with the characteristics of drones, how they operate, and their limitations. Furthermore, future perspectives for drone development need to be considered.

Section 2.1 presents well-known drone classifications in terms of their weight, size, and type. Section 2.2 describes the common hardware and software constituents available on a drone. It is important to understand the interaction between these components to enable the visual guidance of drone-tracker. As a reminder, the drone-tracker is the only drone pursuing the target animal at a given time step of tracking. Finally, Section 2.3 presents some drone limitations and potential solutions.

2.1 Generality

An unmanned aerial vehicle (UAV) is an aircraft able to fly without having a human pilot on-board. It is also referred to by the terms drone, remotely piloted aircraft (RPA), remotely operated aircraft (ROA), or unmanned aerial system (UAS). The latter underlines the fact that a UAV is more than an aircraft. It typically has other components which are primarily the ground control station (GCS) or the command, control, and communications (C3) system and the operator or the command and control link [19].

The aircraft of a UAV can fly either remotely-controlled or autonomously using a predefined path or a given technology [19]. The proposed system is designed to fit the latter since, in order to displace to collect data, a drone-tracker has to utilize the visual information captured by its camera. The intended application falls into the category of

wildlife monitoring which corresponds to civilian use of drones.

Other drone civilian applications are: precision agriculture where UAVs are utilized for crop monitoring and pesticide sprinkling [20], wireless access network, remote sensing, search and rescue, goods delivery/logistics, structural inspection [21], photography, cinematography, hobby uses [22], surveillance, public safety, and civil security [23]. However, some of the civilian applications can also be found in the military domain. Other drone military applications are: missile launching, bomb-dropping, and espionage mission [22].

Regardless of their areas of applications, it is not straightforward to categorize drones as there are several types of drones. In fact, a drone is characterized by diverse attributes such as its size, weight, flight endurance, flight mode to name four. Tables 2.1, 2.2, 2.3 support this fact since they show three different classifications based on drones' weight by [24; 25; 26], respectively.

Table 2.1: Weight-based drone classification by [24].

Class	**Type**	**Weight range**
Class I (a)	Nano drones	W $\leq$ 200 g
Class I(b)	Micro drones	200g < W $\leq$ 2 Kg
Class I (c)	Mini drones	2Kg< W $\leq$ 20 Kg
Class I (d)	Small drones	20 Kg < W $\leq$ 150 Kg
Class II	Tactical drones	150 Kg < W $\leq$ 600 Kg
Class III	MALE[3]/HALE[4]/Strike drones	W> 600 Kg

[3]Medium Altitude, Long Endurance, [4]High Altitude, Long Endurance.

Table 2.2: Weight-based drone classification by [25].

Designation	**Weight range**
Super heavy	W > 2000 Kg
Heavy	200 Kg < W $\leq$ 2000 Kg
Medium	50 Kg < W $\leq$ 200 Kg
Light	5 Kg < W $\leq$ 50 Kg
Micro	W $\leq$ 5Kg

In terms of size, according to [22], drones spread in a spectrum that ranges from large fixed-wing machine to smart dust (SD) as shown in Figure 2.1. The largest aircraft of the

Table 2.3: Weight-based drone classification by [26].

Designation	Weight range
Micro	W < 2 lbs
Mini	2 lbs $\leq$ W $\leq$ 30 lbs
Tactical	30 lbs $\leq$ W $\leq$ 1000 lbs
Medium and high altitude	1000 lbs $\leq$ W $\leq$ 30000 lbs
Heavy	W > 1000 lbs

drone spectrum has wings long as 61 m and weights up to 15 t [27]. Next, are the micro unmanned air vehicle (μUAV), the micro air vehicle (MAV), the nano air vehicle (NAV), the pico air vehicle (PAV) [28]. The smallest machine of the drone spectrum has a size up to 1 mm and weights a minimum of 5 mg [29].

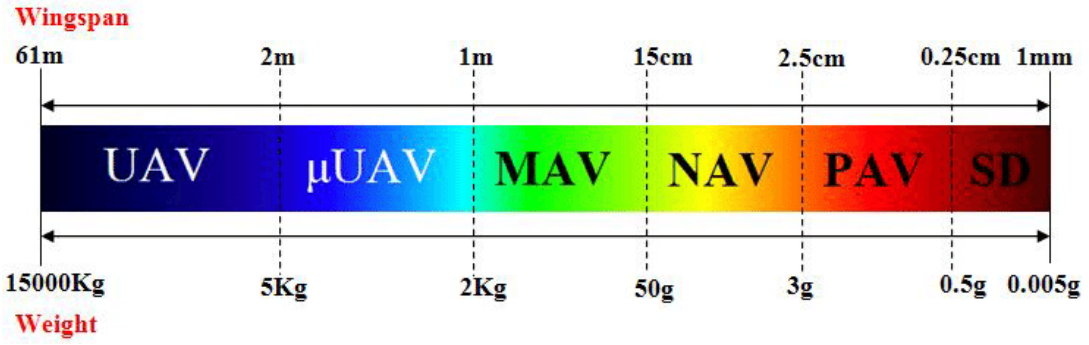


Figure 2.1: A spectrum of drones which spans from vast fixed-wing unmanned air vehicle (UAV) to smart dust (SD) based on wingspan and weight. In between these two types of drones are: micro unmanned air vehicle (μUAV), micro air vehicle (MAV), nano air vehicle (NAV), and pico air vehicle (PAV) [22].

Furthermore, [22] expands the drone spectrum by including all of the existing types of UAVs. This classification is shown in Figure 2.2.

Vertical take-Off & landing (VTOL) drones hover better than horizontal take-Off & landing (HTOL). However, they are restricted in terms of cruise speed. To have UAVs with more capacities, hybrid models were created. The hybrid models are the tilt-rotor, the tilt-wing, the tilt-body, and the ducted fan UAV. UAV helicopters can vertically take-off, land, and hover. UAV helicopters are available in four types which are: single-rotor, coaxial-rotor, tandem-rotor, and quad-rotor. The Heli-wing UAVs have rotating blades and can fly vertically like fixed-wing drones [22].

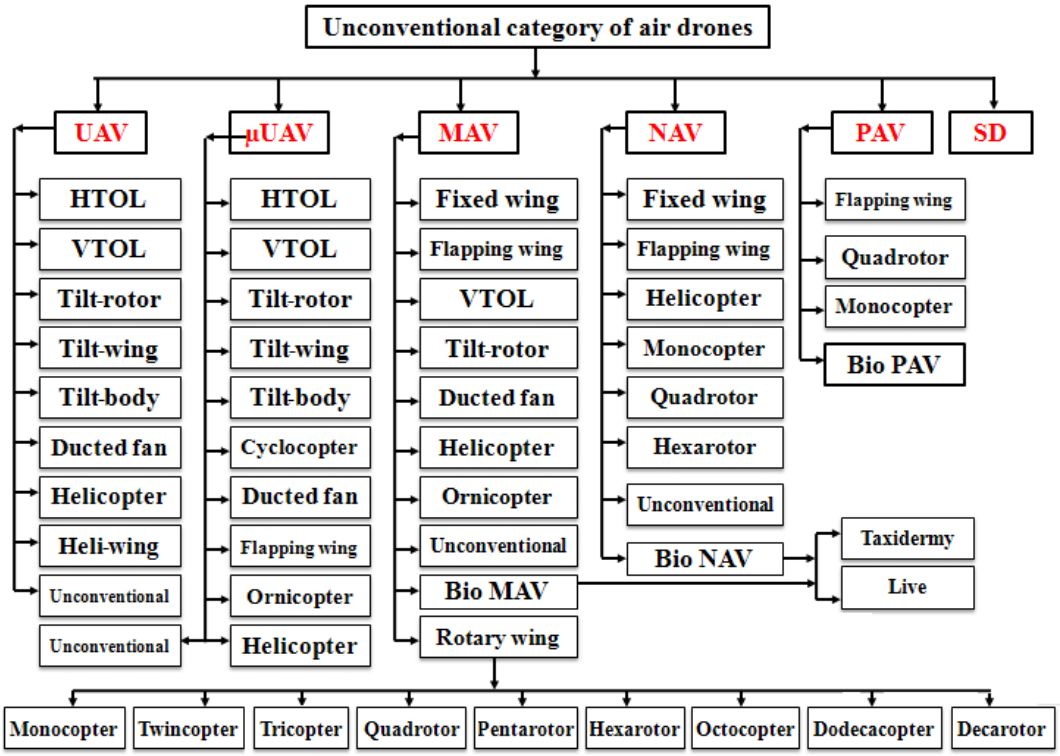


Figure 2.2: Different types of drones [22]. The class of MAV would be suitable for the study because the drones in this class have the ability to hover in a fixed position and fly in all directions [30]. Also, their small dimensions are advantageous for areas that are difficult to access. They can be characterized by the number of motors they involve (one to twelve motors) [31; 32].

The cyclocopters or cyclogyros utilize cycloidal rotors. Cycloidal rotors are made of aerofoils which generate lift and thrust by rotating around a horizontal axis. The ornithopters are UAVs that mimic the opening and closing mechanism of birds' wings when flying. Ornicopters and flapping-wing drones also fly like birds but the former do not have tail rotors while the latter's wings are light and flexible. Fixed-wing drones are generally aircraft equipped with rigid wing, fuselage, and tail [22].

The MAV class includes a subcategory of drones which are highly manoeuvrable since they can hover and fly in all directions [30]. Owing to these advantages and their relatively small size and weight, they might be more suitable than all of the other types of drones for the intended application of this study. These drones are the rotary-wing MAVs which involve rotary blades or propellers. Depending on if they are endowed with one, two, three, four, five, six, eight, ten, or twelve motors they are addressed as mono-copter, twin-copter, tri-copter, quad-copter or quad-rotor, penta-copter, hexa-

copter, octo-copter, deca-copter, and dodeca-copter, respectively [31; 32].

Bio-drones are types of drones which are animal-body based. They are primarily found in the MAV, NAV, PAV classes. They can be taxidermy bio-drones or live bio-drones. In both cases, animals or their bodies are used coupled with electrical components such as sensors [22].

The UAV, μUAV, MAV, NAV categories, all have a subcategory addressed as unconventional drones which groups all of the other machines which are not included in other subcategories [22]. Despite their great diversity, drones are often made with the same basic components.

2.2 Components

This section focuses on the basic elements which compose a drone in terms of equipment and programs.

2.2.1 Hardware

The following picture shows some of the most common elements available on a civilian UAV and in particular a quadcopter since it is prevalent amongst civilian drones.

Frame: It is the aircraft's constituent that gathers and maintains all of the other components of a drone together. The choice of the size and material of the frame is very important. As a matter of fact, these parameters can modify the drone's flying performance. The shape of a quadcopter's frame is a "+" or an "x" [34].

Propeller: Also referred to by the term props, it is a spinning wing which enables the production of the thrust and the torque [35] required to lift and suspend the aircraft in the air. A propeller is generally made of injected plastic and composite [36].

One important parameter which characterizes a propeller is the blade pitch. It is defined as the amount by which a propeller would move through a perfect medium in a single revolution [35]. In other words, it is the distance by which a propeller can move with a unique rotation through the air without being affected by external parameters such as the air density.

Multirotors are equipped with two groups of propellers to cancel the counter-revolution forces. These groups are the clockwise (CW) and the counter-clockwise (CCW) propellers. By dint of its propellers, a quadcopter can fly and have a high manoeuvrability [35].

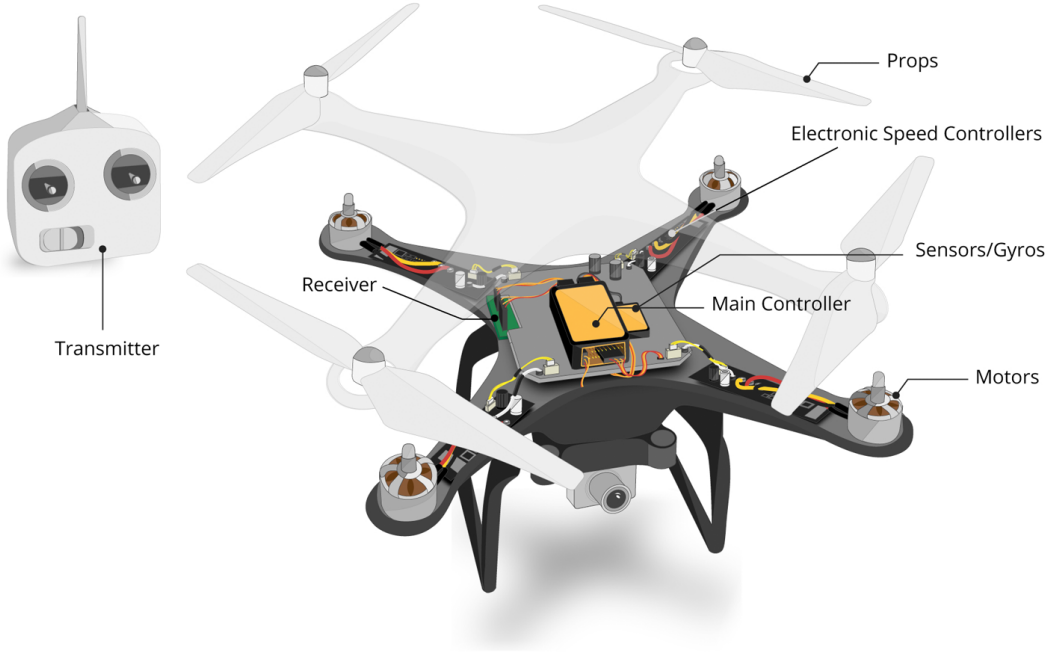


Figure 2.3: Structure of a commercial UAV [33]. Props produce forces for lifting and suspension of the aircraft in the air. They are rotated by the motors which are controlled by the electronics speed controllers (ESCs). The Main Controller initiates and controls the actions of the other components. The Transmitter/Receiver or Transceiver communicates commands and data with the GCS. The drone also embeds a camera. The latter is an essential component for this study as video data are mainly processed.

Motor: The motors are responsible for the rotational movement of the propellers. The number of Revolutions Per Minute (RPM) of an unloaded motor under a tension of one volt is the motor's Kv rating. The higher this parameter is for a motor, the faster the latter turns the propeller. But the lower the motor's Kv rating is, the higher is the propeller torque [37]. For a quadcopter, a motor's size depends on the overall drone weight with its frame size [37] while its type is generally the brushless DC motor (BLDC) [35].

The performance of drone flight can be evaluated with the thrust to weight ratio (TWR). The TWR is the quotient between the thrust of the drone's motors and its weight. A TWR greater than 1 is the value necessary for a drone to take off or to be able to accelerate vertically and a TWR of 1 is the value needed to maintain a drone in the air [37].

Electronic Speed Controller (ESC): It is the drone's equipment that controls the motors' speeds and rotation directions through the modification of their voltages. It is the intermediate between the battery and the motors, while on the other hand, it connects the latter to the receiver [35]. ESCs are essential components for drones which use BLDC motors because they can provide the BLDC motors with a three-phase low voltage power they can produce [35; 38]. Owing to the ESC, multirotor and especially quadcopters can achieve high manoeuvrability as each of their motors can turn at a different speed compared to the other motors [34].

Transmitter/Receiver (Transceiver): The transmitter/receiver pair (transceiver) is crucial for flying the UAV aircraft because it allows the latter to be remotely piloted by an operator. Owing to the transceiver, commands can be sent from the ground station to the aircraft, while telemetry and data such as video can be sent from the aircraft to the ground station through radio signals. The radio frequencies usually used for a drone transmitter/receiver are: 5.8 GHz, 2.4 GHz, 1.2 GHz, or 900 MHz [39].

Sensor: It is a device that can monitor a physical phenomenon and transmit its observation in the form of a signal. Sensors play important roles in flying a drone as well as the task the drone is required for. One of the most common navigation sensors is the inertial measurement unit (IMU). The IMU is a device that can measure an aircraft's orientation, position, and velocity. It can perform these measurements due to other sensors it encompasses. These sensors are primarily the gyroscope, the accelerometer, and the magnetometer.

The gyroscope measures rotational variations of aircraft around the x, y, and z directions induced by forces such as the wind. In other words, the gyroscope records the aircraft's roll, pitch, and yaw, modified by forces that act against the aircraft. These measures are transmitted to the flight controller which utilizes them to steer the aircraft and smooth the flight. Both the three-axis gyroscope and the six-axis gyroscope exist. The latter differs from the former in its additional 3D accelerometer [40]. The accelerometer is a sensor that measures an aircraft's acceleration. Drones also involve other sensors. Often, they are a thermal sensor to measure the temperature, a barometer for the pressure to determine the drone's altitude, a rangefinder that uses sonar or radar to measure the drone's altitude.

More sophisticated drones that possess obstacle avoidance technologies make use of other types of sensors. They can include the stereoscopic sensor, the ultrasonic sensor, and the infrared sensor. The first is employed to capture object shapes in all three di-

mensions. The second and the third measure the distance from the drone to surrounding objects using ultrasonic waves and infrared signals, respectively [41].

Camera: Drones often carry cameras. A camera is essentially used for applications that involve photographic and/or cinematographic abilities. In this case, HD cameras are more appropriate since they can record high-quality images. They are usually coupled with three-axis gimbals to enable the camera's tilt and diminish vibrations caused by the drone's motion [42].

A camera can be used for drone navigation. Indeed, a drone can be remotely piloted by a pilot on the ground (person or a software system) based on images captured by the first-person view (FPV) camera [42]. Furthermore, it exists the thermal imaging camera which can also be mounted on drones [39].

GPS: Drones integrate the global positioning system (GPS), which is based on a satellite navigation system for geographical position establishment [39]. It is often used for predetermination of the waypoints a drone can follow to fly autonomously. However, GPS suffers from privacy concerns and multipath signals or denial of services induced by objects such as tall buildings or tunnels. Also, there is the need to regularly replace the satellites since they have a limited lifespan and are prone to hazardous events that happen in space and expose them to damages[43]. A similar satellite-based global system for localization addressed as GLONASS, or global navigation satellite system is also utilized for drone positioning. Some drones even integrate both systems [43].

Main Controller: The main controller is also addressed as flight controller [38]. It is the primary computational unit of the drone. Indeed, the main controller initiates and coordinates the actions of the other components. It relies primarily on the IMU. For most commercial drones, the flight controller embeds the IMU, as well as other navigation systems such as GPS [42].

Diverse devices and circuits can be used as flight controllers. Some that are often involved are specialized microcontrollers or universal ones such as Raspberry or Arduino boards. A flight controller choice is relative to the complexity of the task the drone has to perform [35].

The main controller can receive commands from human operators via the receiver. The latter is in contact with the transmitter. Programs that aim to permit drone autonomous flight need to integrate this component. On the other hand, the manufactur-

ers of drones update the firmware which can be loaded to a flight controller, primarily using USB interfaces which also serve for drone configuration [42].

Battery: The battery supplies energy to the components of a drone. Some drones, and in particular multirotors, generally employ Lithium polymer (LiPo) batteries which are also largely used in other electronic devices such as smartphones, laptops, and tablets. The two other most widely used alternatives are Nickel metal hydride (NiMH) and Nickel Cadmium (NiCd) batteries. But they have fewer capacities, lower discharge rates, and are heavier than LiPo batteries [44].

The selection of a type of relevant drone for an application can be difficult since drones exist in a large variety and include diverse technologies. For guidance, the articles [42; 45; 46] present reviews of the best civilian drones available on the market regarding their components and capabilities.

2.2.2 Software

Drones use software to enable and optimize the capabilities of their hardware components. The constituents of drones that software developers and manufacturers focus on the most are the main controller, the gyroscope, the accelerometer, and the barometer since they are the main elements required for autopilot of a drone [39]. Autopilot systems have been enhanced in recent years through multiple open-source projects like the Dronecode Project founded by groups and companies such as the Linux Foundation, 3D Robotics, Baidu, DroneDeploy, Intel, JDrones, Laser Navigation, and Yuneec to name a few [47]. Table 2.4 presents some of the open-source drone projects which are used primarily for autopilot but also for other functions.

The source codes of ArduPilot, MultiWii, AutoQuad, LibrePilot are all released under the GPLv3 license, while Paparazzi UAV is under the GPL License. On the other hand, Dronecode, ROS, and PX4 are all three under the BSD license. Despite the availability of such software tools, drones still have some limitations which need to be addressed.

2.3 Limitations and Potential Solutions

Drones have some challenges that hamper their use. Firstly, most drones are restricted in terms of energy level especially when powered by batteries. Secondly, there is a lack of regulations concerning their utilization [56]. Thirdly, drones are potential targets for hackers. In fact, cases have been reported where drones have been used to hack and

Table 2.4: Non-exhaustive list of open-source drone projects.

Project	**Description**
ArduPilot	It possesses both software and hardware products. The ArduPilot's software is advanced, full-featured, reliable, and used by more than one million vehicles ranging from air engines to even boats and BalanceBots [48].
MultiWii	It was initially developed for the gyroscopes and the accelerometers of the Nintendo Wii consoles. It has been later extended to a broader range of sensors and multirotors [49].
LibrePilot / OpenPilot	LibrePilot was created in July 2015 from OpenPilot. One of its main areas of focus is the development of software and hardware for the control and stabilization of robots including flying drones. [50].
AutoQuad	It proposes hardware and software solutions for flight stabilization, dynamic, and autopilot. However, a part of AutoQuad is proprietary. The project is suitable for multicopters that have up to 14 motors. But it can be readjusted for monocopters as well as fixed-wing aircraft. [51].
Dronecode	It was initiated by the Linux Foundation that was later joined by other members. It is a wide open-source project which develops lasting solutions for drones. These solutions are flight control programs, communication system protocols, and safeness solutions, APIs for widespread programming languages to name three [52].
Paparazzi UAV	It was founded in 2003. It has available open-source software and hardware specialized in ground stations and autopilot for a wide range of drones. It can easily be installed and used on diverse operating systems [53].
Robot Operating System (ROS)	It is a multi-purpose framework for the development of portable software for robots [54].
PX4	It is a platform to create innovative and scalable software-based solutions for drone applications, especially autopilot. PX4 is also adapted to platforms such as underwater vehicles and boats. [55].

take control of other drones while flying, by intrusions in their wireless communication systems [42]. Fortunately, solutions to prevent such attacks have been presented in [57]. Fourthly, for noise-sensitive applications as is the case here, they can be difficult to deploy since concerns about disturbances to animals exist. Short-term solutions to this problem could be to fly a drone as high as possible and to avoid drone approaches which could generate strong wildlife responses [58]. Camouflage could also be used to make the drone less apparent to the animals. The study in [58] provides more guidelines to help minimize the potential effects of UAVs on wildlife.

Fortunately, drone development is a very active field of research. Indeed, the drones' limitations and the restrictions regarding their use could be addressed shortly. As a matter of fact, encouraging results for such works are already available. *Energy Or Technology Inc.*, a Canadian company specialized in the development of PEM fuel system demonstrated that multirotors could fly for almost four hours [59]. Furthermore, *MetaVista* a South Korean Company holds the record for the longest multirotor flying time which is 12 hours, 7 minutes, 5 seconds. They achieved this by operating a 6-litre liquid hydrogen cylinder and Intelligent Energy's 800 W Fuel Cell Power Module to power the UAV [60]. Another example of promising accomplishments in drones is the development by the German company *Skysense* of a charging pad which can be used outdoors [61]. This device is particularly interesting for the deployment of the proposed drone formation (Figure 1.1) because it can reduce human inputs and therefore increase system autonomy. Each drone could be charged after a tracking mission once back at its initial position.

The evolution of other technologies can be beneficial for drone development. The technology for printing object in 3D has permitted the rapid fabrication of some drone constituents [56]. The evolution of telecommunication systems has improved the communication of commands to aircraft in terms of rate and distance [19; 62].

In conclusion, the chapter reviewed drones in terms of their diverse attributes (weight, size, type), their hardware and software components, and some of their limitations. Due to this review, a potential category of drones which better fits the proposed animal monitoring system has been identified. It is the class of MAV drones and more specifically the quadrotors which use electric motors. They are relatively small, light in weight, and have some interesting manoeuvrability capabilities.

Quadrotors of the MAV class, that use electric motors are more compatible with the results obtained by [58]. They found that fuel engines might not be adapted for wildlife

studies which involve drones compared to electric engines because they can generate reactions in animals. The aircraft's size can also induce animals' responses since animals can associate a larger aircraft with a threat compared to a smaller one with both aircraft flying at the same altitude. Furthermore, [58] demonstrated that animals' reactions also depend on their species. Indeed, terrestrial mammals are less responsive to drones, compared to birds. Note that the study in [58] has also proposed guidelines for flying drones in a way that reduces as much as possible wildlife disturbance.

The quadrotors offer the benefit of manoeuvrability which is required for tracking animals since the latter's motion can be random. Yet, this ability can affect the video data captured. However, a quadrotor can be equipped with gimbals to help stabilize its camera and facilitate its motion. Moreover, the components of drones operate together for drone flights. The Main Controller initiates and controls the actions of the other components. It relies on the IMU. The ESC defines the motors' movements and speeds. The motors spin the propellers. The battery powers the relevant components. The Transmitter/Receiver communicates with the GCS. They are very important because due to them, flying instructions can be sent to the aircraft while the aircraft can send information back. They represent a potential solution in handling the challenge of drones' payload which prevents drones from carrying heavy computational devices for on-board processing (which also require energy). Additionally, progress in related fields, i.e. telecommunications, could mitigate this limitation.

Chapter 3

Video Object Tracking: A Literature Review

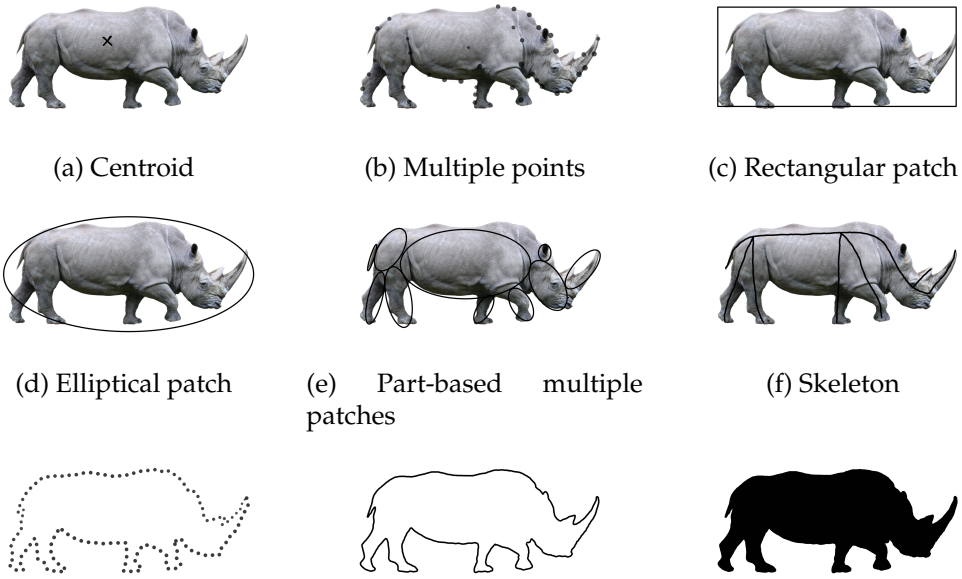
Video object tracking (VOT) has been addressed over the years by researchers due to the myriad of possible applications and the multiple challenges such as changes in illuminations and scale, occlusion, background clutter, low-resolution targets, target deformation, fast motion, and motion blur to name a few [63]. In recent years, the success of deep learning in related computer vision tasks has opened a new venue for object tracking [63]. This chapter reviews the VOT literature in order to appreciate the methods available and select the ones which better suit the present work and its challenges.

Section 3.1 highlights generalities about VOT algorithms. Next, Sections 3.2 and 3.3 present categories of object detection methods and VOT techniques, respectively. The categorization introduced by [64] is followed and extended on the one hand with deep learning-based object detectors (Subsection 3.2.3) and on the other hand with correlation-filter based trackers (Subsection 3.3.4) and deep learning-based tracking algorithms (Subsection 3.3.5). Finally, Section 3.4 discusses studies which are closely related to this project.

3.1 Generality

Video object tracking (VOT) consists of using an algorithm to compute or estimate the position of a given object of interest in successive video frames. It can be applied in areas such as autonomous vehicles, human-machine interactions, traffic-flow monitoring, robotics, activity recognition, security, and surveillance [63]. This last area encompasses wildlife monitoring which is the subject of this study.

VOT systems primarily involve four components which aim to initialize the target, to model its appearance, to estimate its motion and its position [63]. Initializing the target consists of representing it and identifying its location in the first frame. The appearance model represents the tracked object based on its features and the tracking framework. The most widely used object models are a point for the centroid of the object or multiple key points, a simple geometric shape (rectangle, ellipse), the object's contour or its silhouette, or an articulated shape model (see Figure 3.1). On the other hand, mostly used appearance representations are a probability density function, a template, an active appearance model, a multiview appearance model. Some of these appearance models are jointly used with shape models [64]. Motion estimation consists of finding a region where the object is most likely to be in the following frames while the target positioning goal is to precisely locate the object. The latter usually relies on maximum posterior prediction or greedy search. However, the focus is generally given to the appearance and motion model for simplification purposes [63]. Moreover, constraints such as constant velocity or constant acceleration can also be assumed in order to simplify the tracking problem based on prior knowledge (number and size of objects, appearance, and shape, etc.)[64].



(a) Centroid (b) Multiple points (c) Rectangular patch

(d) Elliptical patch (e) Part-based multiple patches (f) Skeleton

(g) Controlled contour points (h) Complete contour (i) Silhouette

Figure 3.1: Object representations, following [64]: the figure above is an adaptation to the context of wildlife tracking.

VOT is subject to multiple challenges which make it a non-trivial task. These challenges are for example the possibility of an abrupt motion, the modification of both foreground and background patterns, as well as foreground occlusion, changes in the scene illumination, the need for real-time processing, the presence of noise in images, and camera motion [64]. VOT can perform single object or multiple objects tracking, with either generative or discriminative methods and online or offline. They can also be based on the use of a correlation filter[63].

Trackers are also characterized by the object's features which help in representing the object in a unique manner. Some trackers utilize a single feature but most combine different types of features. The colour of a tracked object is commonly used. A substantial part of the proposed tracking framework relies explicitly on colour cues since a colour image segmentation algorithm is employed. Other features which are usually involved in tracking algorithms are the edges of the object; the most popular edge detection algorithm being the Canny Edge detector [65], the optical flow which was used by Horn and Schunck [66] as well as Lucas and Kanade [67] and the object texture [64]. Often, VOT algorithms include detection modules which serve to identify the target in the first frame and/or in subsequent frames [64].

3.2 Object Detection Algorithms

The study in [64] classifies object detectors according to their operation mode into four categories. The categories are the point detector, the segmentation, the background modelling, and the supervised classifier. This categorization is extended with the class of deep learning-based methods.

3.2.1 Point Detector

The algorithms in this category spot interest points in images which have the advantage of being invariant to changes in illumination and camera viewpoint [64]. One of the well-known methods in this category is the Harris detector [68]. It is based on the work of Moravec [69]. The Harris detector finds interest points in an image I by using a second-moment matrix M. The latter relies on first-order image derivatives I_x and I_y in x and y directions. The matrix M is computed for each pixel in small windows, and defined as follows:

$$M = \begin{pmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{pmatrix}. \tag{3.2.1}$$

The determinant and trace of M are employed for the Harris response calculation $R = det(M) - k.tr(M)^2$, where k is constant. Finally, non-maximum suppression is used to pick up the optimal values which represent the interest points. The Kanade-Lucas-Tomasi (KLT) [70] uses the same second-moment matrix M to find potential interest points. Their confidences are computed using minimum eigenvalues of M to eliminate candidates with low confidence values. The final step is to eliminate candidates that are spatially close to each other.

Unlike the Harris detector, the scale-invariant feature transform (SIFT) [71] is invariant to affine transformations. Its workflow is composed of four important steps. Firstly, a set of convolved images are created using the input image with Gaussian filters at different scales. Secondly, the previous images are used to obtain the Difference of Gaussian (DoG) images, the maxima, and minima of which are retained as points of interest. Each point's location is updated based on the interpolation of colour values using neighbourhood pixels. During the third step, the potential interest points which have low contrast and the ones located around the edges are excluded. Fourthly, orientations are attributed to the rest of the candidate points using peaks in the histograms of gradient directions in their neighbourhoods [64].

3.2.2 Segmentation

The purpose of image segmentation is to divide image pixels into clearly distinguishable groups. One technique for segmenting an image is Mean-shift clustering in the joint spatial and colour space [72]. The method iteratively updates a mean-shift vector until obtaining a fixed center for a cluster. The mean-shift vector is the vector, the origin of which is the previous cluster's center, while its end is the current cluster's centre.

Another technique is to employ graph cut to segment an image which is represented by a graph G with vertices the image's pixels. Each edge is weighted following the similarity between the colour, the texture, or the brightness of the two pixels (vertices) it links. The weighted edges are pruned to obtain the graph partitions which are disjoint regions in the image [64; 73; 74].

3.2.3 Background Modelling

Some of the methods of this class use background subtraction. It can be defined as a process that models the background in the first or initial frames and searches for changes that represent the movement of the foreground in subsequent frames. The work of [75] was one of the first to make use of background subtraction. They model each pixel

intensity $I(x,y)$ as a Gaussian, i.e. $I(x,y) \sim \mathcal{N}(\mu(x,y), \sum(x,y))$ for which parameters, the mean $\mu(x,y)$ and the covariance $\sum(x,y)$ are derived from sequence of frames. Next, pixels that belong to the foreground are identified using their Gaussian model. Other methods use together with pixel colour, different features for background subtraction. This is the case for [76], which combines texture and colour in 5×5 local windows in images.

Hidden Markov models (HMM) have also been employed for background modelling. Indeed, [77] uses HMM primarily to classify image blocks into background and foreground states.

Background subtraction algorithms are convenient to track objects in videos recorded with static cameras since the background is fixed. In addition, these methods are generally efficient in terms of computational requirements [64].

3.2.4 Supervised Classifier

Some approaches which perform object detection make use of a set of manually labelled instances to automatically learn features that are specific to the object of interest. In other words, such algorithms approximate the mathematical function which maps the inputs to the outputs based on learning instances. Neural Networks [78], adaptive boosting (AdaBoost) [79], Decision Trees [80] and support vector machines (SVMs) [81] are some of the most widely used supervised classifiers for object detection. Their principle is to generate hypersurface to isolate object classes in a high-dimensional space [64].

3.2.5 Deep Learning Based Object Detector

Well-known deep learning-based object detectors in the state-of-the-art are: the region-based convolutional neural network (R-CNN) [14], the fast and the faster R-CNN [82; 83], the single-shot detection (SSD) [84] and the You Only Look Once (YOLO) [85]. The third version of YOLO, YOLOv3 [15] is described in Chapter 5 since YOLOv3 is integrated as a module in the proposed tracking framework. The last two techniques handle object detection as a regression problem while the former essentially rely on image region classification.

All of the previously mentioned networks are based on CNNs which are inspired by the operation mode of the visual cortex of mammalians. CNNs involve convolutional layers also addressed as conv layers. They typically comprise three stages which are the convolution operation, non-linear activation, and a pooling operation. The first stage, i.e. the convolution operation, is described in the following equation where $S(i,j)$ is an

element of the feature map obtained for a two-dimensional input image I and a two-dimensional kernel K:

$$S(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n), \tag{3.2.2}$$

or

$$S(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n), \tag{3.2.3}$$

since convolution is commutative. Note that i, j, m and n are indices of elements in feature map S, input image I and kernel K. Also, in Equation 3.2.3, the kernel K is flipped relative to I. The second stage is employed to fire the relevant neurons in the feature map. An example of an activation function is the rectified linear unit (ReLU). The third stage helps to downsample the feature map by grouping the elements which belong to the same neighborhood by a representative statistic. One example of a pooling method is Max-pooling which uses the element which has the highest value to represent the other elements in the same region of the feature map [86]. For further details about CNNs, the reader is referred to this book [86].

R-CNN has been a game-changer regarding the use of CNN for object detection as it introduced an algorithm referred to as selective search for the generation of image patches more likely to contain the object of interest. Selective search relies on superpixels to reduce considerably the number of image regions compared to sliding window techniques. The selective search produces fewer interest regions than the sliding window technique which produces image patches at almost all of the possible locations to find the object of interest in the input image. The feature vectors produced by CNN are classified with SVM and optimized with bounding box regression. However, despite the selective search, R-CNN was not efficient enough in terms of running speed [87].

The spatial pyramid pooling (SPP-net) [88] attempted to address this challenge by first running a CNN on the entire input image instead of on each patch generated by the selective search. Next, the locations of the image patches (generated in the initial input with the selective search) are exploited to identify the corresponding patches in the feature map produced by the last conv layer. This mapping is performed by the utilization of a pooling technique. Also, to provide to the fully connected layers, fixed-length feature vectors, SPP-net integrates spatial pyramid pooling in between the conv layers and the fully connected layers. Spatial pyramid pooling uses a fixed number of bins for any input image size and applies max-pooling to each bin. Note that the bins also include features obtained from downsampled versions of the feature maps. Yet, SPP-net is not trainable end-to-end [87].

Fast R-CNN solves SPP-net's drawback by introducing a kind of max-pooling gradient computation which is also applied to overlapping pooling regions. This has increased the accuracy of Fast R-CNN in comparison to SPP-net. The running time has also been improved by dint of the Fast R-CNN architecture which has two output branches for classification and localization with regression. This architecture enables training jointly the two branches.

Faster R-CNN is on average 10 times faster than Fast R-CNN. This is due to its small CNN referred to as region proposal network (RPN) which replaces the selective search. Additionally, Faster R-CNN introduced the concept of anchor boxes. They are generated for each potential location of the object to manage changes in scale and aspect ratio. For a given location, a total of 9 anchor boxes are generated based on three different scales and three different ratios ($1:1$, $2:1$ and $1:2$) [87].

SSD detector network also uses anchor boxes with different aspect ratios. It learns the off-set of a box instead of the box. Similar to SPP-net, SSD processes the entire input image at once to produce a feature map which is next convolved with a 3×3 kernel to provide bounding box coordinates with their class probabilities. It also handles object scales well due to its conv layers, which downsample the input image at different scales [87].

3.3 Object Tracking Algorithms

This section presents methods used for object tracking. These methods can be categorized into point, kernel, and silhouette trackers according to [64]. This classification is adopted, and further extended with two categories which are the correlation-filter (CF) based trackers and the deep learning-based trackers because both have been largely used in recent years due to their efficiency in terms of performance and/or real-time capabilities.

3.3.1 Point Tracking

The algorithms in this category use points to represent the object of interest and to find the corresponding points across frames. For this purpose, these algorithms rely either on deterministic or statistical techniques. The former track the points using heuristics to restrict the searching region [64]. One example of a deterministic point tracker is the Median Flow [89].

Median Flow involves the forward and backward (FB) error to detect tracking reliability. For a sequence of $k+1$ images, $S = (I_t, I_{t+1}, \cdots, I_{t+k})$, the computation of

the FB error for a point p_t relies on the forward and the backward trajectories, $T_f^k = (p_t, p_{t+1}, \cdots, p_{t+k})$ and $T_b^k = (\hat{p}_t, \hat{p}_{t+1}, \cdots, \hat{p}_{t+k})$, obtained from tracking p_t forward and backward, respectively. Note that $p_{t+k} = \hat{p}_{t+k}$. The FB error can be computed using the following equation:

$$FB(T_f^k|S) = distance(T_f^k, T_b^k), \tag{3.3.1}$$

where $distance(T_f^k, T_b^k) = \|p_t - \hat{p}_t\|$ is based on the Euclidean distance. For tracking with Median Flow, $k = 1$. At a time step t, some points arranged in a grid fashion inside a bounding box β_{t-1} (containing the object of interest) are tracked forward, then backward with the Lucas-Kanade tracker [67; 70]. Next, each point's error is computed using combined the FB and the normalized cross-correlation (NCC) errors. Finally, half of the points with minimal errors are utilized to estimate the new bounding box β_t, based on their medians in each spatial dimension [89].

To restrict point correspondence, the deterministic approaches exploit constraints on the object and its properties. For example, the object is assumed to have not displaced far from its previous location, to have a maximal speed or to have a direction and speed which have not much changed from previously. Note that the same constraints can also be defined for the statistical techniques which model the object using the state space approach [64]. The Kalman filter and the particle filter (PF) are algorithms largely used in this subcategory. Chapter 4 presents the PF in more details since the PF is part of the proposed tracking framework.

3.3.2 Kernel Tracking

The object of interest can be represented with a primitive region such as a template, a density-based appearance model or a multiview appearance model [64]. A natural way to track an object using its template is to find, in a brute force manner, the location of the region in a frame which matches the most the template. To reduce the computational complexity of this approach a motion model can be used to restrict the search region [90]. While templates are generally obtained from pixel intensities or colour features, they can be replaced by other relevant representations such as mixture models computed from pixels in a basic geometrical shape.

One major limitation of using a simple geometric representation is the possibility of having parts of foreground and background mixed inside the geometric shape. This can be addressed by constraining the geometric shape to lie inside the foreground. Another method consists of modelling colour or texture with probability density functions, in

such a way that more importance are given to the pixels which are more likely to belong to the foreground.

Multiview appearance models are sets of tracked object representations under various transformations. They are generally learnt offline using classifiers such as SVM. Indeed, these methods acquire the ability to perform a binary classification of image regions during training on positive and negative patches [64]. These patches are usually extracted from the previous target's location and its neighbourhood. The work in [91] proposes for object tracking, an on-line version of the discrete AdaBoost [92] for feature selection. The technique relies on weak classifiers, selectors and strong classifiers. The first are basic classifiers able to produce weak hypotheses h^{weak} which correspond to features. The seconds are mechanisms which choose the best h_{weak} based on the error associated to them during training. The third are weighted average of selected weak classifiers. At each iteration of the algorithm, the diversity of the weak classifiers is increased by replacing the worst weak classifier with a new randomly chosen classifier. On the other hand, to accelerate the process, all of the selectors are sequentially updated based on the importance λ of a single learning sample and a unique pool of features. In addition, [91] shows how important it is for tracking success to use relevant features such as Haar-like features, orientation histograms and local binary patterns (LBP) [93] to represent the object of interest. Techniques used to generate the weak hypotheses are threshold and Bayesian based methods as well as nearest neighbour. But other learning methods can also be used.

Similar to the previous tracker, MILTrack [94] uses Haar-like features to represent the object of interest as well as a version of AdaBoost for on-line appearance model learning. Indeed, it also relies on the computation of a strong classifier from weak classifiers. However, the optimization criterion for the selection of the best weak classifier is different. In addition, MILTrack trains on bags, i.e. sets of instances instead of individual instances. The optimal weak classifier is the one which minimizes a loss function which takes as parameters the log likelihood of bags which are expressed in terms of their instances using the noisy-OR (NOR) function. This function has the merit of giving high probability to a bag if the probability $p(y|x)$ of one of its instance is high, with y the unknown binary label for a patch x. For tracking, at each time step t, a bag is defined by $X^s = \{x|s > \|l(x) - l^*_{t-1}\|\}$, with $s, l(x)$ and l^*_{t-1} a radius in the current frame where patches are cropped, the location for the patch x and the tracker's location in the previous frame, respectively. The new estimated position is determined with the following equation:

$$l^*_t = l\left(arg\max_{x \in X^s} p(y|x)\right). \tag{3.3.2}$$

Note that $p(y|x)$ is to be learnt during training. Next, the appearance model is updated with one positive bag $X^r = \{x|r > \|l(x) - l_t^*\|\}$ and several negative bags containing each, a single negative instance randomly chosen from $X^{r,\beta} = \{x|\beta > \|l(x) - l_t^*\| > r\}$, with $r < s$ and β another scalar. The set $X^{r,\beta}$ contains image patches selected from an annular region defined by the variables r and β.

Tracking-learning-detection (TLD) [95] has three modules as hinted by its name. The tracking module is based on Median Flow [89]. Its errors are corrected with the detection module which also finds in frames processed so far all appearances similar to the target. The learning module which is a semi-supervised learning method identifies the detector's errors in order to avoid them for subsequent tracking. For this purpose, it trains a classifier on a set of samples firstly initialized with labelled data. Next, the trained classifier is used on a set of unlabelled samples. The learning module integrates two subcomponents, the P-expert and the N-expert, to identify samples (false negatives and false positives) that have been wrongly classified and change their labels (into positive and negative) to augment the labelled set. At the following iteration, the classifier is retrained with the new labelled dataset. P-N learning is an interesting approach as it increases simultaneously the classifier generality and discriminability.

3.3.3 Silhouette Tracking

The shape or the contour of the tracked object can be modelled in more details. Therefore, algorithms that employ silhouette tracking have the advantage of handling object shape variations. They operate by either finding in the current frame a silhouette that matches the object model in the previous frame or by evolving a shape from an initial contour using state-space models or objective functions.

Silhouette matching can be performed on the basis of computed distances between a given silhouette and silhouettes detected in a frame. Note that the candidate silhouettes can be identified using background modelling as discussed in Subsection 3.2.3. Silhouette matching can also be performed by determining the dominant flow vectors of pixels inside a silhouette [96] or by using optical flow vectors computed inside silhouettes [97]. The minimization of the energy function for the shape evolution is usually achieved by using gradient descent [98; 99] or greedy techniques [64].

3.3.4 Correlation-Filter Based Trackers

These trackers model the target's appearance with the CF. More recent CF-based trackers have the advantage of running in real-time because they make use of the fast Fourier

transform (FFT) and its inverse, the inverse FFT (IFFT) to perform calculations which are costlier in the spatial domain compared to the Fourier domain. Figure 3.2 shows the general workflow of such trackers. Usually, the CF is initialized with the region centred at the target's location in the first frame. At a given time step, a patch at the target's position which was estimated by the tracker (at the previous time step) is extracted from the current image. A cosine window is employed to smooth the patch. The FFT is applied to the smoothed patch and the CF in order to produce the response map or correlation map. Next, the confidence scores are obtained by applying the IFFT to the response map. The use of the IFFT enables to come back to the spatial domain where the location of the highest confidence score is selected as the new target's position. Finally, features are extracted to update the CF [63]. Generally, the computational complexity of CF-based trackers can be as good as $\mathcal{O}(P \log P)$, where P is the number of pixels in the input. However, they face challenging situations such as the selection of representative features for the target, the latter loss, and changes in scale since the CF size does not change over time [63].

Two of the well-known CF-based tracking algorithms are the minimum output sum of squared error (MOSSE) filter [101] and the kernelized correlation filter (KCF) [102]. Both methods optimize an objective function, although it is performed in the frequency domain for MOSSE while for KCF the objective function is initially in the spatial domain. Equations 3.3.3, 3.3.4 show these objective functions for MOSSE and KCF, respectively:

$$\min_{H*} \sum_i |F_i \odot H^* - G_i|^2, \tag{3.3.3}$$

$$\min_h \sum_i (p(f_i) - g_i)^2 + \lambda \|h\|^2, \tag{3.3.4}$$

where a sample, its label, and a filter are represented by f_i, g_i and h in the spatial domain and their equivalents in the frequency domain, F_i, G_i and H, respectively. The pointwise multiplication and the conjugate operators are represented by $\odot$ and $*$. In the case of KCF, the function p is to be learnt during training while λ is a regularization parameter. Note that MOSSE also employs a regularization parameter for its minimizer. MOSSE and KCF differ in many ways. For example, for training, MOSSE generates eight negative versions of the initial target patch using affine transformations at the initial iteration while KCF performs training on a circulant matrix where rows correspond to cyclically shifted versions of the target patch.

A more recent CF-based tracker is the discriminative correlation filter with channel and spatial reliability (CSR-DCF) [103]. It also follows steps similar to the ones presented in Figure 3.2. But it introduces novel concepts, the channel, and spatial reliabilities.

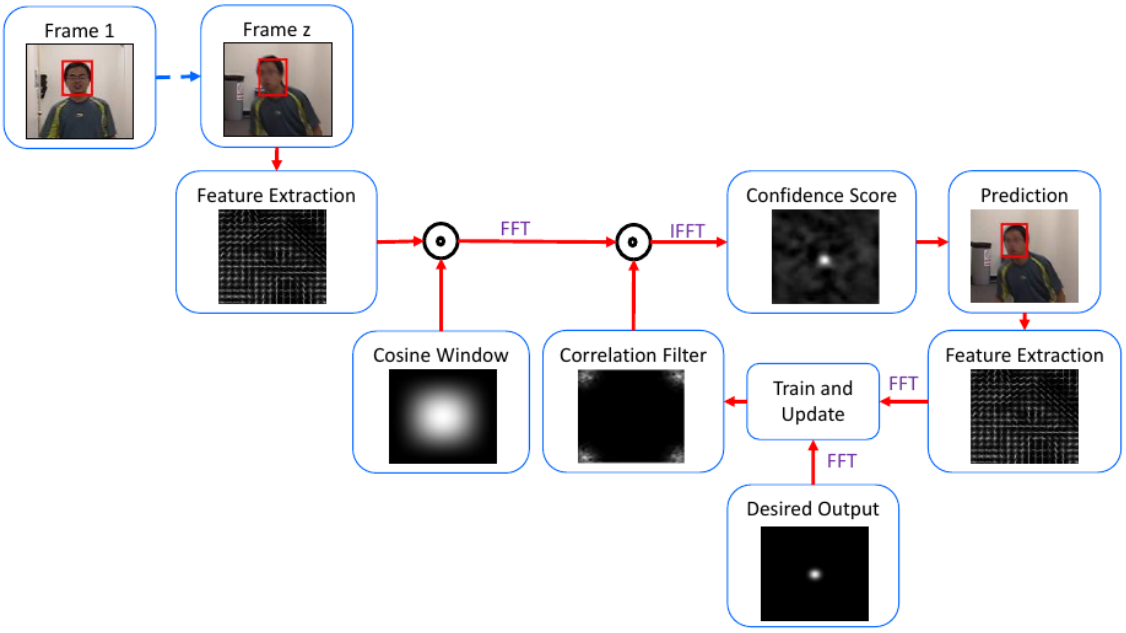


Figure 3.2: General framework for the correlation-filter (CF) based trackers [100]. The CF is initialized with a cropped image of the target in the initial frame. At a time step during tracking, an image patch (or a specific type of feature) is extracted from the previous target's location. The image patch is smoothed with a cosine window, and element-wise multiplication is performed with the CF in the Fourier domain to obtain a feature map. The IFFT is applied to the feature map to produce the confidence map where the peak value corresponds to the new estimated location for the target. The CF is updated with the target's features extracted from the new location [63].

The purpose of the spatial reliability map is to adapt the filters to object parts with the most relevant features. The filters are optimized using the spatial reliability maps which basically are segmentation masks. They are obtained by solving in each frame a graph-labelling problem. The channel reliability scores are used to compute weights for the per-channel filter responses in order to refine the object of interest localization. They address the problem of per-channel feature dominance and are obtained by multiplying the channel learning and detection reliabilities. The CSR-DCF uses HoG and colornames features.

3.3.5 Deep Learning Object Tracking

Some of the methods in this category explicitly extend object detection to object tracking. This is the case for recurrent YOLO (ROLO). Indeed, ROLO achieves online object

tracking by detecting with YOLO the target and saving its location across frames with an LSTM. It is one of the simplest tracking frameworks which operates with LSTM [104].

Generic object tracking using regression networks (GOTURN) [12] is able to track online at 100 fps on average on GPU. This is primarily due to its ability to handle object tracking as a regression problem instead of a classification one and to the use of intensive offline training. By means of offline training, GOTURN can capture specific relationships between objects' appearances and motions. This enables in particular tracking generic objects. The tracker is initialized with the target's ground truth in the first frame. In subsequent frames, a cropped region containing the target at time $t-1$ is coupled to the frame at time t to perform the tracking (Figure 3.3). Therefore, GOTURN exploits information about the tracked object in a given frame to find it in the following frame. This probably explains why GOTURN is able sometimes to handle occlusion. Yet, GOTURN can fail with long-term occlusions. In order to handle long-term occlusions or large movements of the target, the authors suggested the use of a detection module trained online, similarly to what is achieved with the framework of TLD [12].

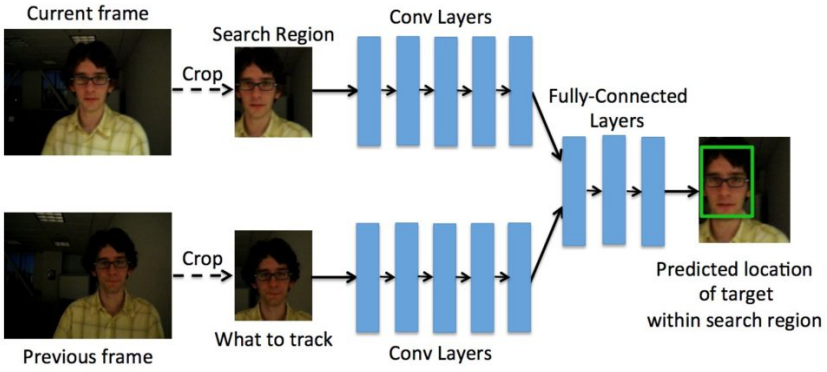


Figure 3.3: GOTURN architecture. The network receives as inputs two image patches. They are obtained by cropping a search region in the current frame and by extracting from the previous frame a region containing the target. GOTURN learns to compare these patches in order to find the target in the current frame [12].

Another deep learning-based tracker is the multi-domain network (MDNET) [11]. It captures a generic object representation by dint of offline training on multiple video sequences. Additionally, MDNET employs a bounding box regression technique and a hard negative mining strategy. The MDNET architecture (Figure 3.4) is composed of shared layers (convolutional and fully connected layers) and K branches of domain-specific layers which correspond to the K training videos. Each domain-specific layer

performs a binary classification to discriminate the background from the foreground in its video or domain. During test on a new video, the K domain-specific layers are combined to create a unique layer which is then tuned online together with the shared fully connected layers [11].

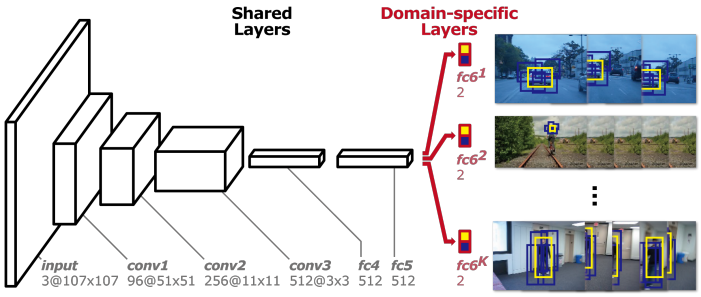


Figure 3.4: MDNET architecture. It is composed by the shared layers and the domain-specific layers. The shared layers capture generic object representations during learning process. The last fully connected layers correspond to the K domains (training sequences). They perform binary classifications. The yellow bounding boxes are the positive samples and the blue bounding boxes are the negative samples [11].

3.4 Related Work

UAVs have already been involved in wildlife conservation through multiple studies and projects. In fact, an inexpensive UAV called Conservation Drone was developed for surveying and mapping forests and biodiversity [7], while the wildlife conservation challenge (wcUAVc) is currently in development for drone-based prevention of wildlife trafficking [105]. Following the same trend, other studies have not only made use of UAVs but also of CV techniques [2; 106] such as optical flow [107] for wildlife monitoring and preservation.

Unlike the above-mentioned methods, some previous studies have utilized explicit tracking algorithms such as the PF for animal tracking and have also applied the SSIM index. Indeed, in order to update each particle's weight, [108] used as likelihood, an exponential function which takes as parameters the dissimilarity index [109] between a reference region associated with the object and a region centred at a particle's position. However, the computational cost of their approach increases significantly with the number of particles. To mitigate this, [110] has proposed a differential SSIM formula.

The use of a variant of the SSIM measure, the complex wavelet structural similarity (CW-SSIM) [111], has been proven to be efficient for automatic tracking reset in an ul-

trasound liver long sequence [112]. Furthermore, the SSIM measure has been used by [113] to compute a binary image to represent the difference of two Landsat images.

Our work is similar to the tracking approach developed in [18]. However, the two studies differ in these ways:

- they used the SSIM index to measure the resemblance between the current frame and a vehicle detection frame to either track vehicles with CSRT, or to update the vehicle detection frame and detect new objects using Faster R-CNN. In this study, the SSIM index is utilized to evaluate the resemblance between the tracked animal, i.e. an ROI surrounding it in a frame and a template of this animal (its initial appearance) to relay a colour segmentation approach with YOLOv3 and vice versa. Therefore, in their case, the SSIM does not monitor the changes in the target's appearance. Rather, it monitors the changes in the whole scene to avoid duplicated detections and to focus on new objects. If there is a change in a particular vehicle's appearance the SSIM cannot *tell*, because it is applied to the whole frame. However, the SSIM can detect that there are no changes for all of the appearances of the vehicles present in the scene if the traffic is static. For this project, when the SSIM value is less than a set threshold for consecutive frames, YOLOv3 also behaves as a tracker due to the presence of the PF, while in their case, the vehicle detection frame is continuously updated, Faster R-CNN keeps detecting vehicles without tracking them with CSRT.

- The PF and the anchor are employed. The anchor is the previous PF's estimate. The PF involves a motion model for predicting the next target's location while the latter helps to restrict the region for searching the target. In [18] there is no explicit motion model. The only component which makes use of the target's trajectory in their tracking framework is implicitly integrated to the tracker involved in their work, i.e. CSRT.

- The intended application aims to track animal motions which are less predictable than on-road vehicle motions [107].

- Here, the SSIM index is also required to validate the animal identity during the handover of a tracking.

YOLO has been involved in other VOT applications. A first example is [114], where YOLO has been used as real-time multi basketball player detector and simple online and realtime tracking (SORT) [115] as tracker. A second example is [116], where features of

shallow YOLO (S-YOLO) have served to train an individual tracker based on random ferns (RF) [117] in order to estimate the likelihood of the PF particles.

In summary, this chapter presented object detectors, VOT algorithms and works which have common features with this study. With this succinct survey, concepts and techniques suitable for the chosen platform, i.e. drone for wildlife monitoring, have been identified. Examples of concepts are assumptions on the animal's motion smoothness and its speed to simplify the tracking task. The techniques selected are the PF, the image segmentation based on colour and the YOLOv3 deep learning algorithm.

Colour image segmentation is computationally efficient as it simply replaces in the image of interest pixels' intensities by either 0 or 1 depending on how they compare to a set threshold. Therefore this method is suitable for aerial robots like MAV drones which cannot embed heavy and very fast computing devices for processing images on-board. This also justifies the choice of YOLOv3 as YOLOv3 is one of the fastest deep learning methods available nowadays. On the other hand, YOLOv3 is involved to correct the colour image segmentation which can produce many false positives.

The PF framework is suitable for this study because it can estimate the target's position from visual information since one of the key feature of this study is the unavailability of wearable tracking devices. Also, it can handle noisy measurements which are expected with the colour image segmentation as well as images from camera sensors which are inevitably noisy. Moreover, the target's motion is prone to abrupt changes. The PF can handle all of these situations as it deals well with the uncertainties in observations and model when estimating the target's position [64].

Well-known trackers implemented in OpenCV contribution version 4.1.0 were described. Such a description of these trackers is relevant in this study because these trackers are used in Chapter 5 to label video sequences, to analyse how they work relative to the proposed VOT approach and even as the second measurement provider for the VOT solution. The trackers are Boosting, CSR-DCF (also addressed as CSRT), MIL, KCF, GOTURN, MOSSE, median-flow and TLD. They are based on machine learning, correlation filters, deep learning and heuristics. Some of them also make use of features such as orientation histograms, Haar-like features, local binary patterns and/or colornames to model the target.

Chapter 4

Proposed Animal-Tracking System

This work aims to employ aerial robots with VOT algorithms in order to facilitate wildlife preservation. A system consisting primarily of four components is proposed. The components are a drone formation, the proposed tracking framework, the handover between drones, and the approach to update the drone-tracker's position. As a reminder, the drone-tracker is the current drone following a target amongst all of the other drones in the formation.

The first component is required because several drones are needed to compensate for the limited battery life of a single civilian drone in the case of relatively long tracking. Therefore, only one drone is allowed to perform tracking at a time. The formation takes the shape of a static virtual grid where drones occupy strategic positions to ensure coverage of the habitat of the animals under observation. In this work, the observed animals are primarily taken to be terrestrial animals. Since the estimation of animal trajectories is important both for the study of the animals' behaviour and for the drone-tracker's visual guidance, a framework for VOT is designed. It is one of the main contributions of this study. The tracking framework combines a simple and real-time tracker, the colour image segmentation technique, with a more sophisticated but slower method, the deep learning object detector YOLOv3, for single-animal tracking. The two techniques are used to furnish observations to the particle filter (PF) with turns triggered by the SSIM index between the initial and the current target's appearance.

Furthermore, the SSIM measure is used for the validation of the target's identity during the relay or handover between the drone-tracker and the drone-partner (selected beforehand by the drone-tracker). Handover permits the system to benefit from having multiple aerial robots. Its success relies on the knowledge that the two drones have about each other. Therefore, a communication protocol for the handover is also de-

signed. It takes the form of two algorithms that describe the actions of the two drones. The reader is invited to consider these algorithms as well as other ones. All of the algorithms are presented in Appendix B for clarity. The outputs of the proposed VOT framework are utilized to generate periodically the pitch and yaw required for the update of the drone-tracker's position, when the drone-tracker is not hovering above the target.

Section 4.1 explains the configuration chosen for the drone formation, the reasons for this choice, and the assumptions made, to name three. The actual VOT solution is then developed in Section 4.2. Next, Section 4.3 presents the handover process between two drones. Finally, Section 4.4 presents the solution designed to address the drone's visual guidance based on the outputs of the proposed VOT framework.

4.1 Drone Formation Configuration

For the drones to alternately collaborate to perform animal tracking, the study has introduced a static formation where each drone has a particular position when not flying. The drones would be on the ground and get scrambled when needed, as opposed to that they hover in place. Depending on the shape of the grid that defines a formation, the number of drones involved can change. Additionally, to simplify the complexity of such a tracking system, several assumptions are made.

4.1.1 Configuration Choice

The general idea is to use a grid configuration to divide the area that the drones are to monitor and navigate. The grid is composed of cells, where each cell is under the surveillance of one or several drones to compensate for the limitation of a civilian-drone battery. The drones primarily carry cameras to track animals and collect visual data. The cameras are also needed to control drones displacements during tracking to avoid the use of invasive tracking devices.

Recall that the drone formation retained for this study is the one presented in Figure 4.1. With this configuration, it is possible to monitor a region, hereby a cell, by more than two drones. Its choice followed the process of development and analysis of other prototype systems. These configurations are depicted in Figures 4.2 and 4.3, where there is one drone per cell and a drone placed at the intersection of four cells, respectively. Yet, compared to the selected configuration (Figure 4.1), they present some drawbacks.

In the configuration in Figure 4.2, there is no possibility of handing over a track within a cell. Also, for a drone-tracker to perform handover outside its cell, there is a lack

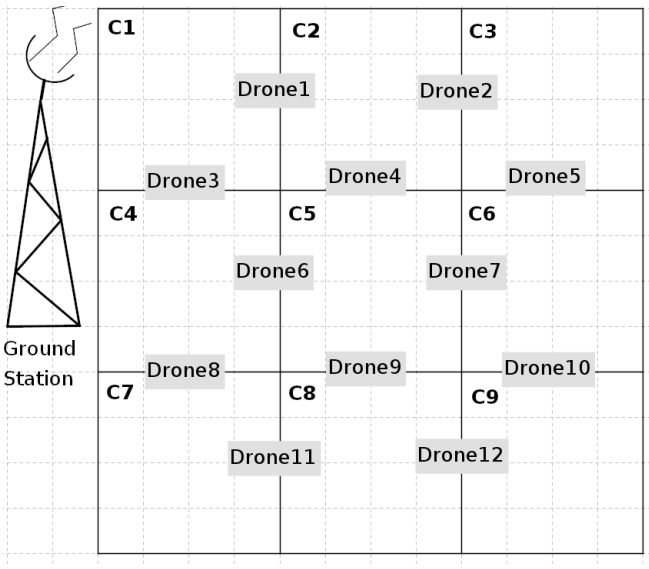


Figure 4.1: Drone formation that involves a drone at the boundary between two adjacent cells. The present configuration illustrates the case where there are 12 drones, in a 3×3 arrangement. The grid has 9 cells labelled C1,C2,...,C9.

of alternatives if the initially chosen drone-partner has a breakdown. These drawbacks are mitigated for the configuration in Figure 4.3, but the drones involved should have

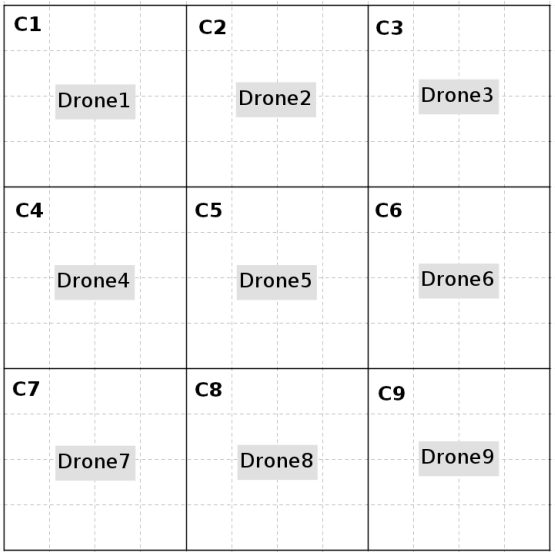


Figure 4.2: Drone formation that involves a drone per cell. The present configuration illustrates the case where there are 9 drones, in a 3×3 arrangement. The grid has 9 cells labelled C1,C2,...,C9.

relatively high battery lives compared to those of the other configurations. The selected configuration (Figure 4.1) solves all of the drawbacks presented by the configurations in Figures 4.2 and 4.3, but more drones are required. The formation includes a ground station (GS) to handle some tasks listed in what follows.

4.1.2 Number of Drones in a Formation

The formation presented in Figure 4.1 has 12 drones dispersed on top of a 3×3 grid (9 cells). However, depending on the grid dimensions, the number of drones varies. This number is defined by N_{Drones}. The purpose of this subsection is to establish and want to establish a formula to determine N_{Drones} in the general case of a $W_G \times H_G$ grid.

To have the possibility of handing over tracking in this type of formation, it is required to have $N_{Drones} \geq 4$, $W_G \geq 2$ and $H_G \geq 2$. In fact, with only one drone in a unique cell, there is no other drone available for handover. Similarly, with either $W_G = 1$ and $H_G > 1$ or $W_G > 1$ and $H_G = 1$, handover is not an option for the single drone responsible for each of the cells at the extremities of the row-grid and column-grid formations.

In these conditions, if the interior of a row in a grid is considered, i.e. without its upper and lower borders, there is $W_G - 1$ drones since each drone is placed at the common-border of two cells. Therefore, all H_G rows in the grid have $H_G(W_G - 1)$ drones. By

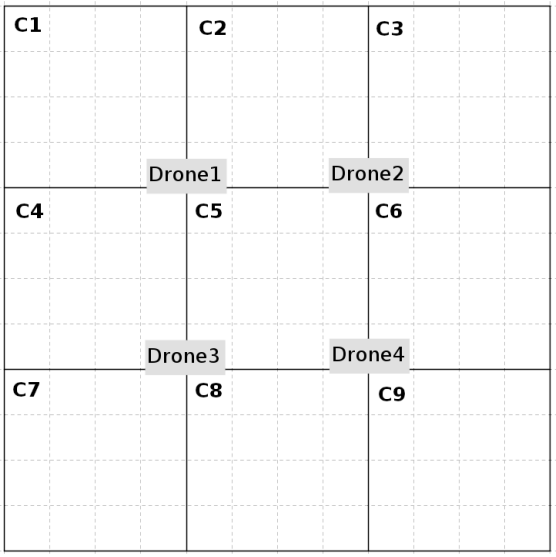


Figure 4.3: Drone formation that involves a drone at the intersection of four cells. The present configuration illustrates the case where there are 9 drones, in a 3×3 arrangement. The grid has 9 cells labelled C1,C2,...,C9.

similar reasoning, for a column in a grid without the left and right borders, there is $H_G - 1$ drones and then $W_G(H_G - 1)$ drones for all of the columns. Hence, N_{Drones} can be computed as follows:

$$N_{Drones} = H_G(W_G - 1) + W_G(H_G - 1) \qquad W_G \geq 2, H_G \geq 2. \tag{4.1.1}$$

In the particular case of a square-grid formation i.e. $H_G = W_G$, the previous equation becomes:

$$N_{Drones} = 2H_G(H_G - 1) \qquad H_G \geq 2. \tag{4.1.2}$$

The variable N_{Drones} is not expressed as a function of the number of cells N_{Cells} in a grid because it is possible to have two formations with the same N_{Cells} and different N_{Drones}. For example, this is the case for a 3×4 grid formation and a 6×2 grid formation which have, both 12 cells but 17 and 16 drones, respectively. The previous example shows that one might have to consider the shape of the target animals' living area, to decide on the grid formation to adopt.

4.1.3 Drone Modes

The condition of a drone referred to in this study as mode, can change during the tracking of an animal. The different modes identified for a drone are described as follows:

- **Sleep** This mode is the condition of a drone that is grounded at its initial position as opposed to flying. The propellers of a drone in this mode are not spinning. Also, a drone in this mode can communicate with either the GS or the other drones. It can also be charged using an outdoor charging pad like the one mentioned in Section 2.3. A practical consideration is that drone on *Sleep* mode can be damage by animals. To prevent this scenario, underground stations could be constructed depending on how feasible it would be in wildlife reserve or park.

- **Track** This mode defines the condition of a drone that is currently tracking an animal. Only one drone at a time can be in this mode during the progress of a tracking.

- **Handover** This mode describes the condition of a drone selected for handover. It also corresponds to the condition of a single drone at a time.

- **Upload** This mode is the condition of a drone which is sending tracking videos to the GS for storage.

- **Return** This mode is attributed to a drone which is returning to its initial position after a handover failure or after a tracking mission.

Some of these modes are exclusive compared to the others. These are the mode *Track* and the mode *Handover*. At a time step, they cannot be taken by more than one drone in the formation and cannot be associated with any other mode. However, a drone could be in the mode *Upload* and in either the mode *Sleep* or the mode *Return* simultaneously.

4.1.4 Ground Station

A ground station (GS) is included in the system to handle some tasks during the global tracking algorithm. These tasks are defined as follows:

- **Tracking initialization** A human operator can use the GS to select the initial drone-tracker and assign it the relevant information about the animal to monitor, such as its location.
- **Videos Storage** The GS also stores the tracking videos recorded during tracking missions and uploaded by the drones.
- **Tracking Termination** The GS could be used to monitor tracking duration (or any other parameter) to terminate an instance of tracking. However, this could also be implemented with drones to reduce the GS involvement in the formation. It would preserve the distributive aspect of the latter.

4.1.5 Model Assumptions

The monitoring system and the simulation of its components can be simplified with some assumptions. A subject for future work would be to conduct sensitivity analysis on the main assumptions of the system. Some of these tests would require that the system be implemented in practice.

- It is assumed that a human operator initiates tracking using the GS to choose and fly the first drone-tracker near the selected target. One can think of a collaborative automatic search process to locate the initial target. An alternative would be to involve a trained ML technique to detect the target in the first frame. However, tracking initialization is not addressed in the present work.
- During the *Sleep* mode, a drone can also charge its battery using an outdoor charging pad (Section 2.3). This assumption is important because all drones need to

have a minimal energy level to be able to communicate with either the GS to upload the previously-captured data or with the drone-tracker in the case of a handover.

- It is assumed that drones maintain the same altitude and orientation during handover. The drone-peer can make suitable data corrections relative to the information it would receive from the drone-tracker during handover. Also, the GPS or any other global navigation satellite system (GNSS) utilised by the drones has enough coverage, especially during handover.
- A drone-tracker's maximum speed is higher than the maximum speed of the animal to be tracked. This assumption is important because the drone-tracker should be able to move rapidly in the field to maintain the target in its FOV in case the target reaches its maximal speed.

The formation described above is not by itself sufficient to solve the tracking task. In fact, the drone-tracker needs a VOT solution that fits its limited computational power to be able to obtain the target's position estimate in the video frames captured.

4.2 VOT Algorithm

This section presents the components of the proposed VOT framework. These components are the PF, the colour image segmentation, the SSIM index, and the YOLOv3 object detection algorithm. Note that a detailed algorithm (Algorithm 2) for the implementation of the proposed VOT framework is available in Appendix A.

4.2.1 Particle Filter

First introduced in 1993 as Bayesian bootstrap [17] and also referred to as sequential Monte Carlo, the PF is an algorithm used to estimate a state vector of a dynamical system which cannot be directly measured. This makes it suitable for the study since the visual information is required to estimate the animal's location for drone guidance. As a reminder, the study is concerned with single-animal tracking in video frames. Note that the Kalman filter can also be used. Yet, the use of the PF offers more flexibility for replacing the components of the proposed VOT solution, and for the extension of the latter to multi-animal tracking. The PF can handle multi-modal densities, which would be expected for multi-animal tracking.

Algorithm 1 describes the steps of the PF for the proposed VOT solution. The particles represent hypotheses of the state vector $x_k \in \mathbb{R}^n$ to be estimated at each time step k. The state vector is assumed to change over time following: $x_{k+1} = f_k(x_k, s_k)$, where $f_k : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is the state transition function and $s_k \in \mathbb{R}^m$ a zero-mean white noise process. Here, x_k corresponds to the actual x and y coordinates of the animal in a video frame. In other words, the target's position is estimated in pixels in a Cartesian coordinates system. Each particle is associated with a weight that represents how likely the particle is to be the actual state vector. The weights are obtained using measurements $y_k \in \mathbb{R}^p$ which are related to the state vector via the observation equation $y_k = h_k(x_k, \nu_k)$ with $h_k : \mathbb{R}^n \times \mathbb{R}^r \to \mathbb{R}^p$ the measurement function and $\nu_k \in \mathbb{R}^r$ another zero-mean white noise process [17].

Algorithm 1 Particle Filter

1. *Initialization:* The initial particles $\{x_0^i | i = 1, 2, \cdots, N\}$ are drawn randomly from a PDF p_0 which is a discrete uniform distribution in this work. Therefore, each particle has an initial weight of $1/N$.

2. *Prediction Update:* At a time step k, the particles are updated using as motion model a random walk:
$$x_k^i \leftarrow x_{k-1}^i + v\Delta T + \eta_k^i. \tag{4.2.1}$$
In this equation, v is the speed of the target animal, ΔT is a small-time constant between two frames and η is a zero-mean Gaussian white noise used to spread the particles. This motion model is used because it is assumed that the animal moves smoothly between two frames. However, this model does not capture changes in the animal's speed and orientation. So an area for further work would be modelling the dynamics of the actual animals' motion; zebras' motion might differ from cheetahs' motion, etc.

3. *Measurement Update:* During this step, a new measurement y_k is obtained using h defined as follows:
$$y_k = h(\{c_k^j\}) = \underset{c_k^j}{\arg\min}\{\|\hat{x}_{k-1} - c_k^j\|\}, \quad 1 \leq j \leq \tau, \tag{4.2.2}$$
where $\{c_k^j\}$ is the set of the target's potential positions while τ is its cardinality. The elements c_k^j are either centroids of segmented regions when obtained from the colour segmentation approach (Subsection 4.2.2) or bounding box centres when

obtained from YOLOv3 (Subsection 4.2.4). The variable τ can be reduced in the case of the colour segmentation approach with a defined threshold to exclude extremely small segmented areas. The previous estimate of the target animal's position $\hat{x}_{k-1}$ is utilized to eliminate the other hypotheses given by any of the measurement providers. It is referred to in what follows as the *anchor*.

The measurement y_k is used to update the weights of the particles with the likelihood function which is defined by: $w_k^i = w_{k-1}^i p(y_k|x_k^i)$. The following equation shows how p is computed:

$$p(y_k|x_k^i) = \frac{1}{\|y_k - x_k^i + \epsilon\|}, \tag{4.2.3}$$

where $\epsilon \ll y_k - x_k^i$. Next, the weights are normalized and the animal's location in a video frame is estimated using this equation:

$$\hat{x}_k = \sum_{i=1}^{N} w_k^i x_k^i. \tag{4.2.4}$$

4. *Resampling:* During this step, N particles are chosen with replacement based on their probability w_k^i. This step is to mitigate sample impoverishment that happens because the set of particles loses its diversity after several iterations. The selected particles' weights are set to be the same, i.e. $1/N$. This step is performed only if the following condition is satisfied:

$$\hat{N}_{Eff} = \frac{1}{\sum_{i=1}^{N}(w_k^i)^2} < N_{thresh}, \tag{4.2.5}$$

with $\hat{N}_{Eff}$ an approximation of the effective sample size and N_{Thresh} a threshold value [118; 119; 120].

5. *Iteration:* The algorithm is iterated by applying: $k \leftarrow k+1$. Therefore, the steps are repeated from item 2. The algorithm stops when the halting condition arises. For example, this condition can be the animal's arrival in a particular cell or a running-time limitation.

4.2.2 Colour Image Segmentation

The colour image segmentation is employed to provide a measurement y_k to the PF because the colour image segmentation can run easily on computationally-limited devices

such as drones and still perform well in general. It consists of obtaining a binary image $\mathbf{X}'$ from an image source $\mathbf{X}$ by thresholding pixel intensity values:

$$\mathbf{X}'(i,j) = \begin{cases} Val, & \text{if } \mathbf{X}(i,j) > thresh \\ 0, & \text{Otherwise.} \end{cases} \tag{4.2.6}$$

The variables i, j represent the pixel locations, Val is generally the maximum pixel intensity value in the colour space considered and $thresh$ is the threshold pixel intensity value.

4.2.3 Structural Similarity Index

The SSIM index [16] is a full-reference image quality assessment metric. The term full-reference refers to the availability of the original image (not distorted). However, this study is concerned with using the SSIM index for change detections, in the animal appearance for the single-animal tracking algorithm and to confirm the target animal's identity during handover between drones (Section 4.3).

The SSIM index can better evaluate the difference between two images relative to other metrics such as the mean squared error (MSE) or the peak signal-noise ratio (PSNR). This is because the SSIM index utilizes the structural information from images as the basis for comparison, while the other methods rely on pixel-wise error.

The intuition behind using the SSIM measure as a change detector is that the target animal's appearance at a time step should still exhibit similarity with its initial appearance despite changes that modify the latter. Such changes can be caused by illumination variation or occlusion by another animal or object, to give two examples. Indeed, it is proposed to evaluate the resemblance between the region of interest (ROI) containing the target animal in the current frame and a template of the target animal using the SSIM index, the simplest version of which can be computed with the following equation:

$$SSIM(\mathbf{x}, \mathbf{y}) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \tag{4.2.7}$$

where $C_1 = (K_1 L)^2$ and $C_2 = (K_2 L)^2$ with $K_1 \ll 1$, $K_2 \ll 1$ and L the dynamic range of pixel intensity values (255 for 8-bit grayscale images) [16]. The coefficients C_1 and C_2 are used to stabilize the SSIM index. The variable $\mathbf{x}$ is the reference image signal with mean intensity μ_x and standard deviation σ_x while $\mathbf{y}$ is the distorted image signal with mean intensity μ_y and standard deviation σ_y. The covariance between $\mathbf{x}$ and $\mathbf{y}$ is given by σ_{xy}. Note that $\mathbf{x}$ and $\mathbf{y}$ are assumed to be non-negative discrete signals and that μ_x, σ_x, and

σ_{xy} are computed as follows:

$$\mu_x = \frac{1}{P}\sum_{i=1}^{P} x_i, \tag{4.2.8}$$

$$\sigma_x = \left(\frac{1}{P-1}\sum_{i=1}^{P}(x_i - \mu_x)^2\right)^{\frac{1}{2}}, \tag{4.2.9}$$

$$\sigma_{xy} = \frac{1}{P-1}\sum_{i=1}^{P}(x_i - \mu_x)(y_i - \mu_y), \tag{4.2.10}$$

where x_i is the i^{th} element of the discrete signal $\mathbf{x}$ and P is the total number of elements in $\mathbf{x}$.

The SSIM index can be computed in local windows in images instead of globally for the quality maps to have locally isotropic property. It is then advisable to use an 11×11 circular-symmetric Gaussian weighting function $W = \{w_i | i = 1, 2, \cdots, P\}$ which has standard deviation 1.5 samples and weights normalized [16]. The following equations express estimates of local statistics μ_x, σ_y and σ_{xy} in this context:

$$\mu_x = \sum_{i=1}^{P} w_i x_i, \tag{4.2.11}$$

$$\sigma_x = \left(\sum_{i=1}^{P} w_i (x_i - \mu_x)^2\right)^{\frac{1}{2}}, \tag{4.2.12}$$

$$\sigma_{xy} = \sum_{i=1}^{P} w_i (x_i - \mu_x)(y_i - \mu_y). \tag{4.2.13}$$

Next, the overall SSIM index is evaluated by computing the mean SSIM (MSSIM) as follows:

$$MSSIM(\mathbf{X}, \mathbf{Y}) = \frac{1}{M}\sum_{j=1}^{M} SSIM(\mathbf{x}_j, \mathbf{y}_j). \tag{4.2.14}$$

The variable M is the number of local windows, $\mathbf{x}_j$ and $\mathbf{y}_j$ are the j^{th} local window in images $\mathbf{X}$ and $\mathbf{Y}$, respectively [16].

For this application, the SSIM value of the relevant ROI in the current frame and the target animal template is evaluated at each time step k of the PF. Hence, when this value is above a certain threshold, the resemblance between the ROI containing the animal appearance and the template is strong enough for the PF to get a reliable measurement from a method with a low computational cost such as a colour segmentation approach. This is of great advantage for this application as the tracking system is required to run in real-time. However, when the SSIM measure is below the threshold value, the similarity

between the target and the template is too weak for a colour segmentation approach to provide effective measurements. A more robust method is then needed. YOLO is chosen, in particular, YOLOv3 [15] as it performs well in real-time relative to other methods in the same class.

4.2.4 You Only Look Once Object Detector

YOLO [85] is a deep learning-based algorithm which uses convolutional neural networks (CNN) to perform object detection in real-time. YOLO baseline and fast YOLO, a lighter version, run at 45 and 155 frames per second, respectively (on a Titan X GPU), while still maintaining good performance[5]. Moreover, YOLO is highly generalizable compared to other state-of-the-art detectors such as deformable parts models (DPM) [121] and R-CNN [14] since it better deals with unexpected inputs [85].

YOLO achieves this trade-off between speed and performance because it handles detection as a regression problem. In fact, YOLO uses the full input image at once to simultaneously predict bounding box coordinates and class probabilities. In contrast, the other methods apply classification to input image patches. Additionally, by using the entire image during the training and testing phases, YOLO implicitly captures contextual information such as the background compared to the other methods.

The initial version of YOLO has difficulty localizing small objects [85]. However, YOLOv3 [15] has helped to address this challenge by predicting boxes at three different scales (Figure 4.4). Although this has resulted in speed slow-down, YOLOv3 is still faster than most deep learning-based object detectors. For example, YOLOv3 can process on GPU, a single 320×320 image at 22 ms, which is 3 times faster than SSD, with both methods performing on par [15].

YOLOv3 is the version employed in this work. Despite the fact that it is not the best in terms of accuracy (compared to other state-of-the-art object detectors), its capacity to run fast in real-time is of fundamental importance for this study. Having YOLOv3 miss detections in some frames should not affect the proposed tracking algorithm, because other components could compensate. In fact, the PF always estimates a location for the animal since it uses the motion model to move the particles, although the weights are not updated without a new measurement. On the other hand, the colour segmentation method can relay YOLOv3 when the SSIM index is above the set SSIM threshold.

As seen with the proposed VOT solution, addressing the single-drone tracking problem involves elements such as the YOLOv3 object detector and the SSIM index. The

[5] On PASCAL VOC 2007, 63.4 mAP for YOLO baseline and 52.7 mAP for Fast YOLO [85].

latter can also serve to evaluate the similarity between the target representations in both FOVs of a drone-tracker and a drone-relay in order to confirm the target's identity.

4.3 Tracking Relay Between Drones

One main contribution of this work is the tracking relay between a tracker, *DroneA*, and a peer, *DroneB*, during handover using an image similarity evaluation to validate the target's identity. The SSIM index is also employed as an image similarity measure for handover. Prior to and during handover, *DroneA* and the selected *DroneB*, activate their GPS (or another GNSS, preferably with a small location error). This is primarily for *DroneA* to start a handover process based on its position or to enable the displacement of the selected *DroneB* at the specific location of the handover. Note that in practice, due to the requirement of accuracy for drone location, differential GPS (DGPS) would be more appropriate to the system than standard GPS.

This section firstly presents the mechanism by which *DroneA* selects a *DroneB* amongst drone candidates. Secondly, this section describes the handover process. Thirdly, it is addressed in this section the design of a communication protocol between *DroneA* and

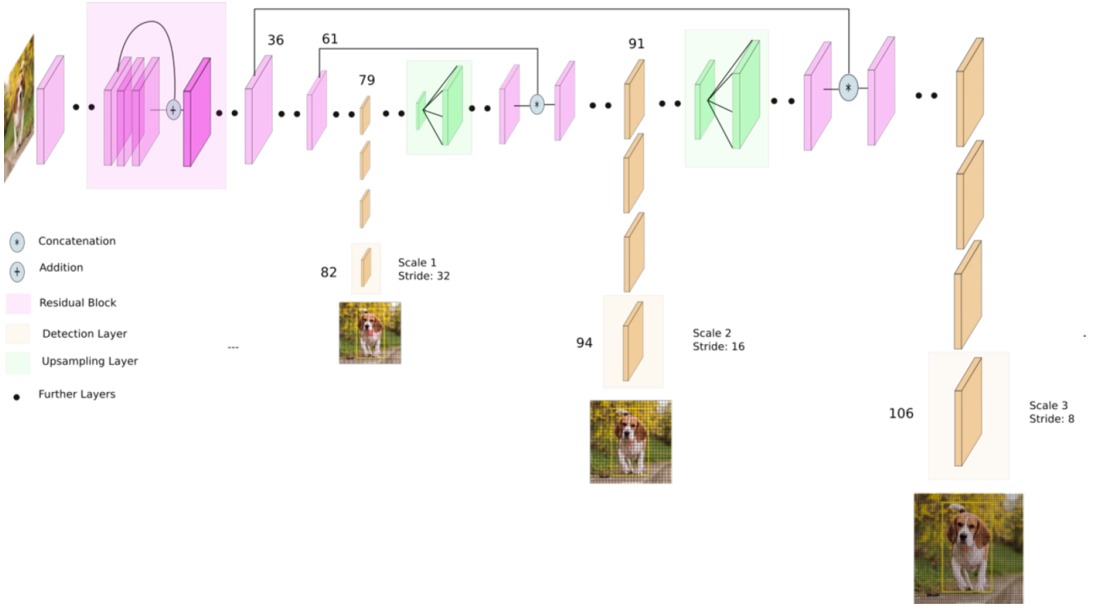


Figure 4.4: The architecture of YOLOv3 [122]. The residual blocks are used to skip several layers to provide further layers with earlier features. Strides are values used to downsample the dimensions of the images.

the current *DroneB* since a handover's outcome depends on the reliability of the exchange of messages.

4.3.1 Drone Handover

Handover happens when *DroneA* is running low on energy level or when it arrives in a cell monitored by other drones. The choice of *DroneB* to take over the tracking is made by *DroneA* based on the proximity of available drones, their energy levels, and even data storage capacities. This choice is made through a mechanism presented in the next subsection. Once *DroneB* is selected, it joins *DroneA* following location information received from the latter in a way that their fields of view (FOVs), FOV_A and FOV_B, align as shown in Figure 4.5.

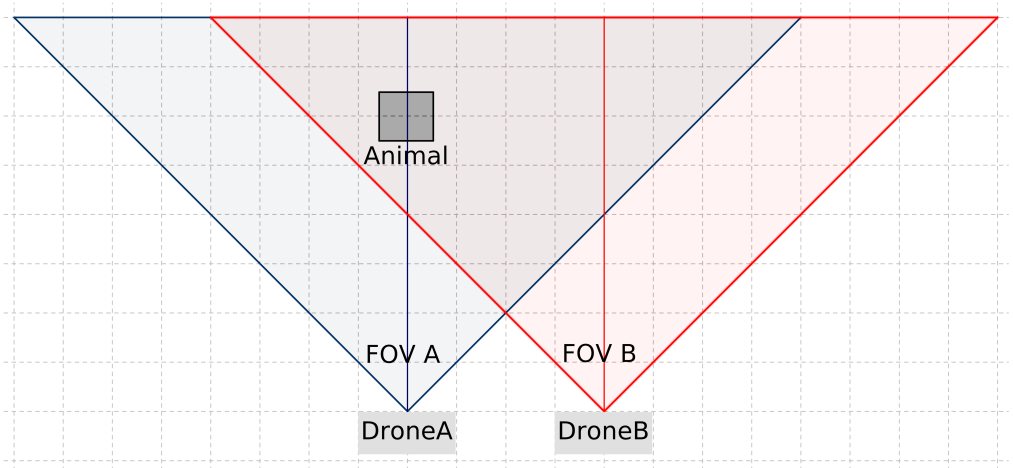


Figure 4.5: Drone alignment for handover.

This side-by-side drone configuration is important to the overall tracking algorithm since to track, drones rely on visual information since the target animal is not fitted with a tracking device. It is assumed that all of the drones in the formation are of the same type, i.e. have the same characteristics. Also, there is the assumption that the drones involved in a handover process maintain the same altitude and orientation during handover. Indeed, *DroneB* would have to adjust its altitude and orientation to the altitude and the orientation of *DroneA* during handover. Therefore, the position of the target in FOV_A will be shifted by a certain amount in FOV_B with respect to the main axis. This amount which should be proportional to the real distance between the drones can be empirically determined in practice.

The GPS signal coverage is assumed to be strong enough to allow the two drones to come into alignment as shown in Figure 4.5. This implicitly means that the location selected for handover is important. However, in case the GPS signal is not reliable during handover, one can think of augmenting it with another signal such as radar. The handover steps are described by Algorithm 3 (Appendix A). It is assumed that drones are always able to send/receive messages, messages do not get lost and *DroneB* has enough energy level to continue the handover process.

4.3.2 Partner-Drone Selection Algorithm

For a drone to be in the mode *Handover*, it has to be selected by the current drone-tracker which is in the mode *Track*. At each time step of a track, *DroneA* has to check its energy level and position in order to decide whether to continue the tracking or to call another drone-relay. The selection of the potential *DroneB* is based on several parameters which are the drones available, their energy levels, their distances to *DroneA*, and their capacities in terms of data storage. Algorithm 4 (Appendix A) shows the steps *DroneA* has to follow to choose and call another drone for a handover.

4.3.3 Drones Communication Protocol During Handover

Handover relies significantly on the messages communicated between *DroneA* and *DroneB* since these messages acquaint each drone with the state of knowledge of the other and therefore define what action each drone has to take next to ensure handover success or to avoid dramatic consequences for the whole system in the case of handover failure. This section presents a communication protocol between *DroneA* and *DroneB* for a handover success in general but in particular to avoid the tracking of a target by more than two drones if a handover fails.

This situation is closely related to the well-known *Two Generals' Problem* [123; 124] which is also addressed as the *Coordinated Attack Problem* or the *Two Armies Problem*. In brief, the problem concerns two army generals whose only chance to overcome a common enemy is to mount a simultaneous attack. But, they need to agree on a time by means of messengers since they are located far from each other. They can only reach a consensus by exchanging an infinite number of messages, because the army enemy could intercept a messenger (at any time of this infinite exchange) and avoid both army allies to reach a consensus, exposing each of them to the risk of attacking alone. The problem has been proven unsolvable [123; 124]. The point is the problem is about

achieving common knowledge through asynchronous communication by distributed processors.

The *Two Generals' Problem* is related to the *Byzantine Generals Problem* [125] which involves more than two armies and the possibility of a small number of generals to be traitors. In the context of a distributed network of computing systems, the *Byzantine Generals Problem* refers to the possibility that some of the nodes of this network send conflicting information to the other nodes [125]. Yet, in recent years, the *Byzantine Generals Problem* has been addressed in practice by the Bitcoin Blockchain technology [126].

The two drones represent the army allies while the animal represents the army enemy (although here the goal is in the favour of the animal). There is a slight difference in the consensus to reach between the two drones. In this study, they have to *agree* on the drone which has to continue the tracking and avoid the case with both drones tracking the target. An analogy can be established between the fact that none of the generals is a traitor (only for the *Two Generals' Problem*) because they have primarily agreed on attacking the army enemy, and the fact that the purpose of the handover between the two drones is the relay of the tracking of the animal from the drone-tracker to the drone-peer. Recall that the two army allies only need to agree on the time for the attack and similarly that the two drones only need to agree on the information transmitted to each other during handover.

What prevents termination of the message exchange is that there is no point at which common knowledge can be established through this asynchronous exchange, given the possibility of message failure (capture of a messenger). Similarly, for the two drones, there is the possibility that a handover message could be lost. There might then be no drone-tracker if *DroneA* thinks that *DroneB* is tracking when it is not, or there could for a time be two trackers, while *DroneA* tries to call another *DroneB* candidate for another attempt, the previous *DroneB* could also be tracking the target.

One could think of a solution involving Blockchain technology for the communication of the drones during handover. However, this solution would require to set a limit for the number of messages to exchange between the two drones, since there is still no guarantee that a message is received after it has been sent. On the other hand, the time that a single transaction requires in the Blockchain technology is not suitable for the real-time requirement of this study. In addition, even if the time requirement could be met, using this technology might require additional hardware facilities since drones' computational capabilities are limited. The handover communication protocol relies on an asynchronous approach that integrates realistic concepts such as the possibility of a message being lost and the fact that message exchange cannot be infinite.

From now on, notations are borrowed from the work of [127] which contributes to epistemic modal logic, to formulate the problem of message exchange between *DroneA* and *DroneB*. Let ρ represent the knowledge that encompasses all useful information to be transmitted by *DroneA* to *DroneB* after both drones have aligned for handover. The variable ρ stores information such as the target's position estimate at time step k, the corresponding particles, and weights. The variable ρ can be considered as a shared secret in a modern and secure communication system between two ends. *DroneA* knows ρ is written as $K_A\rho$ using the *knowledge operator*, K [127]. Note that ρ is equivalent to $K_A\rho$ since *DroneA* is the one that has the knowledge ρ at first.

Once the two drones are aligned for handover, *DroneA* has to send the message $K_A\rho$ to *DroneB*. After the reception of this message, *DroneB* acknowledges *DroneA* by sending back $K_BK_A\rho$. Next, *DroneA* acknowledges *DroneB* by sending back the message $K_AK_BK_A\rho$. *DroneB* can also acknowledge back with $K_BK_AK_BK_A\rho$. This exchange could go on forever to achieve the common knowledge of ρ in the group of the two agents, *DroneA* and *DroneB*. However, it is not possible since for handover to be successful this exchange has to end in order for *DroneB* to relay *DroneA* and for *DroneA* to enter the mode *Return*. Therefore, it is decided to stop the communication between the two drones after three exchanges, i.e. once *DroneA* sends $K_AK_BK_A\rho$ as it is important for the system that *DroneB* knows that *DroneA* received its previous message, is returning home, and is not calling a new *DroneB*. It is also defined three communication levels which correspond to the number of message types allowed.

It is not enough to constrain the number of message levels between the two drones because in practice a message can get lost, not be successfully sent, or received due for example to device failures. Following the previous notation, such cases correspond to $\neg K_A\rho$,$\neg K_BK_A\rho$, and $\neg K_AK_BK_A\rho$. Note that $\neg K_BK_A\rho$ also means that *DroneB* was unable to lock its camera on the target. For these cases, it would be tricky for the drone receiver or sender to know what action it has to take next. It is then defined a time receptive window for message reception, as well as a maximal number of tries for sending a message if there is no response for the previous message. The former is represented by variables Δ_A and Δ_B for each drone, while the latter is represented by the variables n_A and n_B. They have the same values, i.e. $\Delta_A = \Delta_B$ and $n_A = n_B$, but they are differentiated relative to *DroneA* and *DroneB* to avoid confusion. Furthermore, two counters n_{TA} and n_{TB} are involved in counting the number of message-sending tries. A counter ctr_D is dedicated to the maximal number n_D of *DroneB* candidates that *DroneA* can call in case of successive failures.

A new discrete-time step, k' is introduced. It defines the steps of the handover

communication protocol. The variable k' is defined as follows: $k' = k + \alpha\zeta$ with $\alpha = 0, 1, 2, 3, \cdots$ and $\zeta = 1/\zeta'$, $\zeta' \in \mathbb{N}$. It ensures that both drones have the same *FOVs* when ρ is sent from *DroneA* to *DroneB* and when *DroneB* receives ρ from *DroneA*. It is assumed that sending a message and therefore waiting for a message requires each a time step k' while receiving a message and taking an action like entering the mode *Return*, the mode *Track* or calling another *DroneB* (while the previous *DroneB* is returning home) are instantaneous. Hence, $\Delta_A = \Delta_B = \beta k'$ with $\beta \in \mathbb{N}$. In addition, a message emitted at a time step k' can only be received at the following time step $k' + \zeta$. More than one action can be taken at a time step by a drone if there is only one action that is not instantaneous. In practice, ζ has to be tuned in accordance with the duration that sending a message requires. This duration should be smaller than the time needed by a drone to capture a single frame.

In this communication protocol, at any time step k', the first action a drone has to take is to check if there is a new message. Therefore, sending the first message is considered as *DroneA*'s second action while waiting for the first message is *DroneB*'s second action. Without *DroneB*'s second action, the communication has no chance to continue further to lead to the target tracking handover. Hence, a it is imposed to *DroneB* tp wait for a maximum of $n_A(\beta + 1)$ for the reception of a *level* 1 message. This permits to *DroneA* to try n_A times successively to send $K_A\rho$ to *DroneB*, in order for *DroneB* to use ρ to find the target in its FOV. The counters n_{TA} and n_{TB} are initialized to zero. They are incremented by 1 each time the corresponding drone sends a message while they are reinitialized each time the relevant drone receives a different message level. The counter ctr_D is also reinitialized, but it is when *DroneA* calls a new *DroneB*.

Algorithms 5 and 6 in Appendix A show the steps *DroneA* and *DroneB* have to follow, respectively, to decide on their actions at a given time step k' of the handover communication protocol. The latter starts after *DroneA* and *DroneB* have aligned with the first executions of the two algorithms. For subsequent time steps (k'), the drones' algorithms take turns to be executed, unless there is a message arrival. This is due to the assumption about the duration of message-sending compared to its reception which is instantaneous and to the one about a message delivery at the time step $k' + \zeta$ following the one it has been emitted. Therefore, a message reception at arrival is prioritized in this communication protocol. In these algorithms, the instruction *pass* means that the communication protocol has ended for either the previous *DroneA* or the previous *DroneB*. Section 5.3.1 presents simulations of the proposed communication protocol in the following chapter.

Sending and receiving the *level* 3 message, i.e. $K_A K_B K_A \rho$, is determinant for the out-

Table 4.1: Possible outcomes for the handover communication protocol based on the sending, the reception (implicitly the loss) of a message $K_A K_B K_A \rho$. The actions of *DroneA* and *DroneB* in the present table are the last ones both drones can take. Therefore, actions like waiting for a message ($K_A K_B K_A \rho$) or trying to resend another type of message have been performed.

		DroneB receives message $K_A K_B K_A \rho$.	
		True	*False*
DroneA sends message $K_A K_B K_A \rho$.	*True*	• *DroneA* enters mode *Return*. • *DroneB* enters mode *Track*.	• *DroneA* enters mode *Return*. • *DroneB* enters mode *Return*.
	False	Non-applicable.	• *DroneA* calls a new *DroneB* or enters mode *Return*. • *DroneB* enters mode *Return*.

come of the handover. Table 4.1 recapitulates the possible outcomes for a handover. This table confirms that a handover cannot lead to the tracking of an animal by more than two drones as a result of the failure of a handover communication protocol following the non-emission, the non-reception, or the loss of a *level* 3 message.

As the handover is meant for tracking continuation if the previous drone-tracker is no longer able to continue, it shows how important it is to keep tracking the target as long as possible. Therefore, handover contributes to guaranteeing the availability of the target's position most of the tracking duration. It is important that an estimate of the target's position in video frames is available since it is needed for the update of the drone-tracker's position.

4.4 Animal Pose Estimation for the Inference of the Position of the Drone-Tracker

The VOT algorithm developed in this work aims primarily to estimate the animal's position at time step k for the guidance of the drone-tracker. Recall that the study aims to

avoid attaching tracking devices to the target. Indeed, a method is proposed to infer (update) the drone-tracker's position in the real world using the target's position estimates in the video frames captured by the tracker.

Several assumptions are made for simplicity and clarity. The 3D problem is reduced to a 2D problem by supposing that *DroneA* flies at a constant altitude. Note that without this assumption, the target's changes in scale could have been used to control *DroneA*'s elevation.

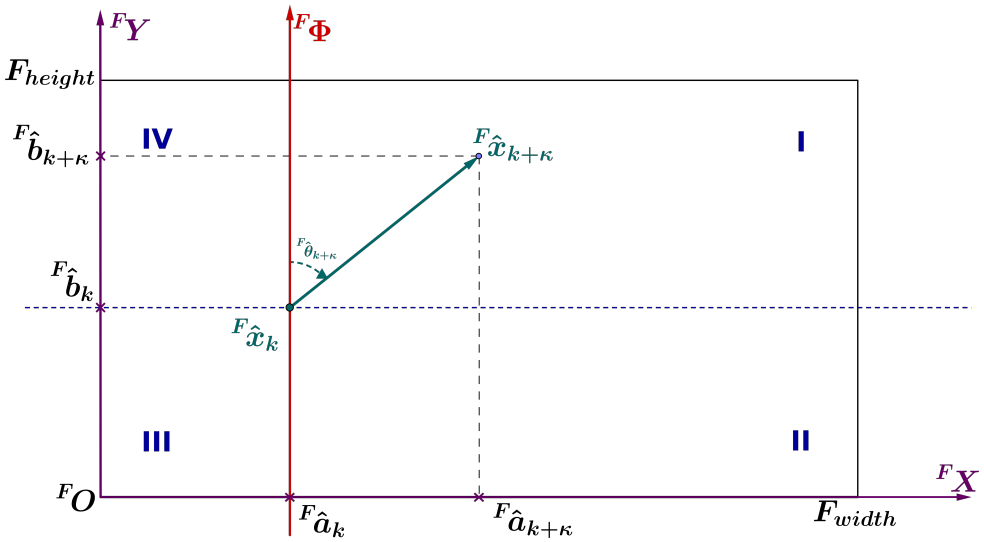


Figure 4.6: Representation of the Field of View FOV_A of *DroneA* with the animal position estimate at time steps k and $k+\kappa$. The prescript F indicates that variables are taken to be in video frames as opposed to the prescript W for the real world. In this representation, a Cartesian coordinate system is defined by the origin ^{F}O, as well as two axes, $(^{F}O\,^{F}X)$ and $(^{F}O\,^{F}Y)$. A polar coordinate system is defined by an origin at $^{F}\hat{x}$ and a vertical axis. The polar axis and the horizontal line passing through $^{F}\hat{x}$ divide FOV_A into four quadrants annotated I, II, III and IV.

Figure 4.6 represents *DroneA*'s FOV between time steps k and $k+\kappa$ during a track. The variable κ is a time-periodicity for updating the drone-tracker's position. Note that in what follows, prescripts F and W are used to indicate that variables are taken to be in video frames or the real world, respectively. Also, the absence of a prescript indicates that a variable is considered in video frames. At time step k, the target's position estimate $^{F}\hat{x}_k$ has as abscissa $^{F}\hat{a}_k$ and ordinate $^{F}\hat{b}_k$ in the coordinate system defined by the perpendicular oriented lines $(^{F}O\,^{F}X)$ and $(^{F}O\,^{F}Y)$. Similarly, at time step $k+\kappa$, the target's position estimate is defined by $^{F}\hat{x}_{k+\kappa}(^{F}\hat{a}_{k+\kappa}, {}^{F}\hat{b}_{k+\kappa})$. However, since in image processing the coordinate system often used has as origin the image's top-left corner and the y-axis

oriented downward, one firstly need to express ${}^{F}\hat{x}_k({}^{F}\hat{a}_k, {}^{F}\hat{b}_k)$ and ${}^{F}\hat{x}_{k+\kappa}({}^{F}\hat{a}_{k+\kappa}, {}^{F}\hat{b}_{k+\kappa})$ as functions of $\hat{x}_k(\hat{a}_k, \hat{b}_k)$ and $\hat{x}_{k+\kappa}(\hat{a}_{k+\kappa}, \hat{b}_{k+\kappa})$, respectively as given by the proposed VOT solution. This is achieved with the following equations:

$$\begin{cases} {}^{F}\hat{a}_k = \hat{a}_k \\ {}^{F}\hat{b}_k = F_{Height} - \hat{b}_k, \end{cases} \tag{4.4.1}$$

$$\begin{cases} {}^{F}\hat{a}_{k+\kappa} = \hat{a}_{k+\kappa} \\ {}^{F}\hat{b}_{k+\kappa} = F_{Height} - \hat{b}_{k+\kappa}, \end{cases} \tag{4.4.2}$$

where F_{Height} and F_{Width} are the FOV_A's height and width, respectively.

The variable κ can be chosen to be equal at least to 1. Such a choice will result in the update of $DroneA$'s position every time step or video frames, captured by its camera. However, updating $DroneA$'s position so often may be too demanding in terms of computational and energetic resources. One can determine an upper bound for κ. In this way, between k and $k+\kappa$, $DroneA$ could hover above (and behind) the tracked animal.

Define v_{Max} as the target's maximal speed in pixels per second. It can be empirically determined or based on the target species' maximal speed. The minimal amount of time T_k^{out} required by the target to go out of FOV_A is given by $\frac{{}^{F}d_k}{v_{Max}}$, where ${}^{F}d_k$ is the minimal distance the target has to travel to exit FOV_A from its position ${}^{F}\hat{x}_k$. The value of ${}^{F}d_k$ can be found with the following equation:

$${}^{F}d_k = \min\left({}^{F}\hat{a}_k, {}^{F}\hat{b}_k, F_{Height} - {}^{F}\hat{b}_k, F_{Width} - {}^{F}\hat{a}_k\right). \tag{4.4.3}$$

Therefore, if γ is the frames rate required by $DroneA$ to capture images, κ's upper bound, κ_{Max}, is given in Equation 4.4.4 as follows:

$$\kappa_{Max} = \gamma.T_k^{out} = \gamma\frac{{}^{F}d_k}{v_{Max}}. \tag{4.4.4}$$

From now, in order to ease the problem of the $DroneA$'s position update, the Cartesian coordinate system is shifted to a polar coordinate system. The latter's has as origin ${}^{F}\hat{x}_k$ and polar axis, the vertical line passing through ${}^{F}\hat{x}_k$ which has the same direction as the y-axis of the previously defined Cartesian coordinate system (Figure 4.6). The angular coordinate is determined clockwise (CW). The purpose here is to compute ${}^{F}\hat{x}_{k+\kappa}$ coordinates $({}^{F}\hat{\phi}_{k+\kappa}, {}^{F}\hat{\theta}_{k+\kappa})$ in the polar coordinate system in order to derive the pitch and the yaw which will permit updating of $DroneA$'s position in the real-world frame.

The radius ${}^{F}\hat{\phi}_{k+\kappa} \in [0, \max(F_{Height}, F_{Width})]$ can be calculated as follows:

$$ {}^{F}\hat{\phi}_{k+\kappa} = \sqrt{\left({}^{F}\hat{b}_{k+\kappa} - {}^{F}\hat{b}_{k}\right)^{2} + \left({}^{F}\hat{a}_{k+\kappa} - {}^{F}\hat{a}_{k}\right)^{2}}. \tag{4.4.5} $$

The angular coordinate ${}^{F}\hat{\theta}_{k+\kappa} \in [0°, 360°)$ can be determined with this equation:

$$ {}^{F}\hat{\theta}_{k+\kappa} = \begin{cases} 0° & if\ {}^{F}\hat{b}_{k+\kappa} > {}^{F}\hat{b}_{k}\ and\ {}^{F}\hat{a}_{k+\kappa} = {}^{F}\hat{a}_{k} \\ 180° & if\ {}^{F}\hat{b}_{k+\kappa} < {}^{F}\hat{b}_{k}\ and\ {}^{F}\hat{a}_{k+\kappa} = {}^{F}\hat{a}_{k} \\ 90° & if\ {}^{F}\hat{b}_{k+\kappa} = {}^{F}\hat{b}_{k}\ and\ {}^{F}\hat{a}_{k+\kappa} > {}^{F}\hat{a}_{k} \\ 270° & if\ {}^{F}\hat{b}_{k+\kappa} = {}^{F}\hat{b}_{k}\ and\ {}^{F}\hat{a}_{k+\kappa} < {}^{F}\hat{a}_{k} \\ \theta & if\ F_{Height} > {}^{F}\hat{b}_{k+\kappa} > {}^{F}\hat{b}_{k}\ and\ F_{Width} > {}^{F}\hat{a}_{k+\kappa} > {}^{F}\hat{a}_{k} \quad (I) \\ \theta + 180° & if\ F_{Height} > {}^{F}\hat{b}_{k+\kappa} > {}^{F}\hat{b}_{k}\ and\ 0 < {}^{F}\hat{a}_{k+\kappa} < {}^{F}\hat{a}_{k} \quad (II) \\ \theta + 180° & if\ 0 < {}^{F}\hat{b}_{k+\kappa} < {}^{F}\hat{b}_{k}\ and\ 0 < {}^{F}\hat{a}_{k+\kappa} < {}^{F}\hat{a}_{k} \quad (III) \\ \theta + 360° & if\ 0 < {}^{F}\hat{b}_{k+\kappa} < {}^{F}\hat{b}_{k}\ and\ F_{Height} > {}^{F}\hat{a}_{k+\kappa} > {}^{F}\hat{a}_{k} \quad (IV), \end{cases} \tag{4.4.6} $$

with

$$ \theta = \arctan\left(\frac{{}^{F}\hat{b}_{k+\kappa} - {}^{F}\hat{b}_{k}}{{}^{F}\hat{a}_{k+\kappa} - {}^{F}\hat{a}_{k}}\right) \qquad \theta \in (-90°, 90°). \tag{4.4.7} $$

Note that for Equation 4.4.6, the first four lines are the cases where ${}^{F}\hat{x}_{k+\kappa}$ falls in the boundaries defined by the vertical and horizontal segments perpendicular at ${}^{F}\hat{x}_{k}$ while the following four lines correspond to the four quadrants defined by these segments and the frame boundaries.

Given these coordinates, i.e. ${}^{F}\hat{\phi}_{k+\kappa}$ and ${}^{F}\hat{\theta}_{k+\kappa}$, *DroneA*'s pitch ${}^{W}\Phi_{k+\kappa}$ and yaw ${}^{W}\Theta_{k+\kappa}$ can be computed with the following equation:

$$ \begin{cases} {}^{W}\Phi_{k+\kappa} = s_{\Phi}\,{}^{F}\hat{\phi}_{k+\kappa} + {}^{W}\Phi_{Comp} \\ {}^{W}\Theta_{k+\kappa} = {}^{F}\hat{\theta}_{k+\kappa} + {}^{W}\Theta_{Comp}. \end{cases} \tag{4.4.8} $$

In this equation, s_{Φ} is a scaling factor to find a real-world equivalent to ${}^{F}\hat{\phi}_{k+\kappa}$ while ${}^{W}\Phi_{Comp}$ and ${}^{W}\Theta_{Comp}$ are constants to compensate the additional target's displacement and rotation during *DroneA*'s position update. Note that ${}^{W}\Theta_{Comp}$ should be very small since it is an angle. Instead of always rotate CW, *DroneA* could also rotate counter CW by an amount of $360° - {}^{W}\Phi_{k+\kappa}$ when ${}^{W}\Phi_{k+\kappa} > 180°$.

To reduce the ranges of ${}^{W}\Phi_{Comp}$ and ${}^{W}\Theta_{Comp}$, *DroneA*'s speed should be tuned to be higher than the target's speed. Additionally, choosing a value of κ smaller than κ_{Max} could help *DroneA* to avoid losing the target after *DroneA* updates its position. The

SSIM measure can also be employed to determine if the second measurement provider, here YOLOv3 has to be called after each *DroneA*'s position update to avoid the target loss. During handover, a value of κ close to 1 is desirable since handover relies on reliable values exchange for the target's positions.

The process above can be repeated to ensure the target's tracking by *DroneA*. It is important to mention that after every κ time steps the polar coordinate system is updated depending on the new animal position in the video frame, in contrast to the redefined Cartesian coordinate system of a video frame. The steps of the drone-tracker's position update is recapitulated in Algorithm 7 (see Appendix A).

Stopping the tracking of an animal can be reduced to stopping the tracker's position update. Therefore, instructions are included in Algorithm 7 (in Appendix A) to terminate a tracking based on time, location, or energy level conditions although a tracking termination can also be the consequence of a handover failure. The location condition can easily be checked relative to the GPS coordinates of a selected halting-cell which should be available since the formation's grid is static.

In summary, this chapter covered all of the substantial components involved in the formation of drones to perform animal tracking. Firstly, are presented: the formation configuration, the relation between its static grid dimensions and its number of drones as well as the assumptions made to simplify the problem. Secondly, it is shown how one can use as VOT solution a PF algorithm that selects measurements from two measurement providers: a colour image segmentation technique and the YOLOv3 object detector. At each time step, the relevant technique to furnish the potential object's position is decided based on the SSIM index value. Thirdly, the problem of drone collaboration is addressing during tracking via the handover process. Finally, a technique that utilizes the target's position estimate to periodically update the drone-tracker's position is proposed. Indeed, the target's positions in the video frames at two different time steps are used to update the drone's yaw and pitch. In between these two time steps, the drone-tracker hovers. Since the system can be implemented in real life, it is important to simulate some of its components which are testable in the absence of physical drones.

Chapter 5

Tests and Results

This chapter contains the results of experiments designed to test some of the components of the tracking system. It is worth mentioning that the system is not implemented as a whole. The components tested and simulated are the proposed VOT solution and both the handover communication protocol and the technique to verify if the target is well relayed between the drone-tracker and the drone-peer.

This part of the study required a considerable amount of time. First, codes have been written, tuned, and debugged for several versions of the proposed tracking framework which are needed to better understand how the proposed tracking framework works in practice. Also, programs have been coded for the simulation of the verification of the identity of the target involved by handover. Furthermore, well-known trackers implemented in the OpenCV contribution version 4.1.0 were employed. They are briefly presented in Chapter 3. These trackers were utilized to label the video sequences of the dataset involved in the study. The outputs of the OpenCV trackers on video sequences were also used for performance comparison with the tracking framework and some of them were further employed to replace YOLOv3. Program running also required some time since averaged results for several executions per tracker were used because some trackers involve randomness and therefore do not always produce the same track.

An analysis is performed for the results obtained from tests with both the OpenCV trackers and the trackers which are based on the proposed VOT solution. Such analysis is important for the interpretation of the results obtained and for the understanding of the reasons why some of the trackers exhibited behaviours that are not necessarily expected. Finally, note that this chapter also provides qualitative results for some of the trackers' outputs to help the reader understand how they operate. The qualitative results are frame sequences extracted from video outputs from single-tracker runs instead

of averaged outputs for reasons which are explained later.

Section 5.1 briefly presents the dataset, the evaluation metric, and the hardware involved in this study. Section 5.2 presents the experiments and results on the dataset. Next, Section 5.3 covers the simulations of the handover on both the communication protocol and the validation of the target's identity for which three techniques are proposed. Finally, Section 5.4 presents a discussion of the results obtained from all of the tests in this study.

5.1 Generalities

This section presents some tools used to perform the tests of the proposed VOT framework and the handover. These tools are the dataset, the evaluation metric, and the hardware. All of the codes used to obtain the results in this chapter are available at this address[6]. For the implementation of the particle filter (PF) and the YOLOv3, the PF module[7] of the *deepgaze* library developed in [128] and this tutorial[8] were followed, respectively. The study involves the programming language Python and the OpenCV library.

5.1.1 Dataset

Methods were developed and tested using a dataset[9] provided by the Max Planck Institute of Animal Behavior (MPI-AB). It contains footage of animals recorded in Kenya with DJI Phantom 4 Pro drones. The main species filmed are zebra (Grevy's and plains), buffalo, and impala. The drones filmed throughout the day, starting at 7 AM and going to 6:30 PM although, the majority of the flights were in the morning.

In general, the drones were flown at altitudes of $60 - 70m$ for smaller species (impala, gazelle) and $80 - 90m$ for larger species (buffalo, zebra) despite that the altitude was adjusted slightly based on the specific situation. The drone's speed was also adjusted, but with the animal's speed. The dataset was initially acquired at 60 frames per second and 4K resolution but later down-converted to 1080p.

6 https://github.com/juliana2535/RM-codes.

7 https://github.com/mpatacchiola/deepgaze/blob/master/deepgaze/motion_tracking.py

8 https://www.learnopencv.com/deep-learning-based-object-detection-using-yolov3-with-opencv-python-c/.

9 For more details about the MPI-AB dataset, the reader is referred to the information available in the *Acknowlegments* (page *vii*).

For testing the implementation of the proposed VOT approach, a video referred to here as the *cows video* was also used. This video is available in the *deepgaze* library. The video presents a challenging occlusion situation of a relatively static cow by a moving cow that passes in front of the former and goes out of the video's frames. The video also contains other cows with one which has a colour similar to the targeted static cow. All cows are grazing. The video viewpoint suggests that the filming camera was fixed. The *cows video* also has a resolution of $1080p$ but a frame rate of 25 frames per second.

5.1.2 Evaluation Metric

To evaluate the proposed tracker on video sequences from the MPI-AB dataset, the precision plot is employed. This choice is due to the nature of the proposed VOT solution which is a point tracker. On the other hand, the study is interested in evaluating the reliability of the trajectories produced by the proposed tracker since the trajectories are important both for the displacement of the drone-tracker and for the comprehension and the interpretation of animals' behaviours based on their trajectories.

The precision plot shows for a video sequence, the ratio of frames where the predicted animal's centre location falls within a certain distance threshold. Note that in this study the ground truth is obtained from some of the trackers implemented in OpenCV contribution version 4.1.0. This error is obtained by computing the Euclidean distance between the estimated centre location and the ground truth. In particular, the ratio of frames with the threshold lower than 20 pixels is reported for each tracker.

For the *cows video* the precision plot is not used because the precision plot does not reflect the trackers' performance. This is due to the specificity of the occlusion case and the cows' scale which is more important than the scale of the animals present in the MPI-AB dataset. For two trackers' estimates, even if both are located on the animals, the distance which separates these estimates is more important than the distance which separates two trackers' estimates for very small targets (i.e. small-scale targets). Additionally, the occlusion happens towards the end of the video. Subsection 5.2.3 explains more on this.

5.1.3 Hardware

The results presented in this work are obtained using a computer with an Intel i7-3632QM CPU with 2.20 GHz frequency per core and 8GB of RAM. However, during code development, resources of the centre for High Performance Computing (CHPC) in South Africa were also utilized.

5.2 Tests and Results for the Proposed VOT Framework

As seen in the previous chapter, the outputs of the proposed tracking framework are important for the operation of the formation of drones. Employing these outputs, the drone-tracker can update its position and keep following the target to monitor the target and to collect video data.

Here, it is presented the image preprocessing steps that were used. Next are presented the results (and their interpretation) on the MPI-AB dataset and the *cows video* for the challenging VOT situation in this video.

5.2.1 Preprocessing

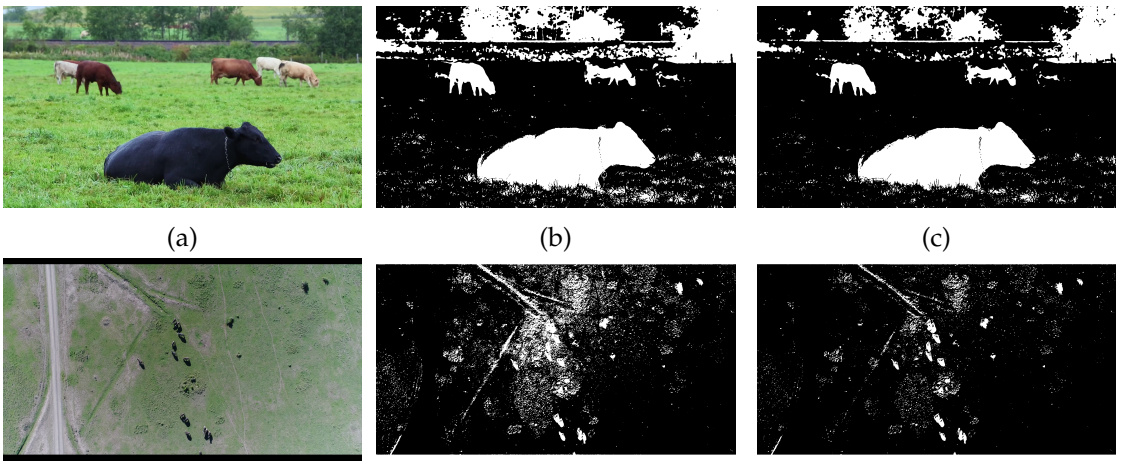
To prepare the frames for the colour image segmentation, the *Gaussian Blur* function available in the OpenCV library was used. This function smooths images, i.e. removes Gaussian noise from images. Note that for the sequences from the MPI-AB dataset, colour image templates that were not necessarily extracted from the given sequence were employed, but they were representative of the animal's colour. Due to the animals' size in these videos, with colour patches extracted from the images, more false positives are observed. It could be the case because the patches do not contain enough information about the targets' colours. On the other hand, for colour threshold for the segmentation, the mean value of the green channel was used instead of the mean of all three channels' averages. Empirically it was noticed that for the video sequences, the former works slightly better than the latter (Figure 5.1) or similarly (Figure 5.2).

For YOLO, the image preprocessing step primarily consisted of scaling the images using the factor $1/255$ to make the range of the pixel values to lie in $[0, 1]$. This is because YOLOv3 has been trained on inputs which are first scaled in the same range of values.

5.2.2 Tests and Results For the Video Sequences Extracted from the MPI-AB Dataset

This subsection considers many practical aspects that have to be addressed in performing the experiments. Firstly, this subsection presents the video sequences, the determination of their ground truth, the introduction of the trackers which involve the PF and the initialization of the trackers.

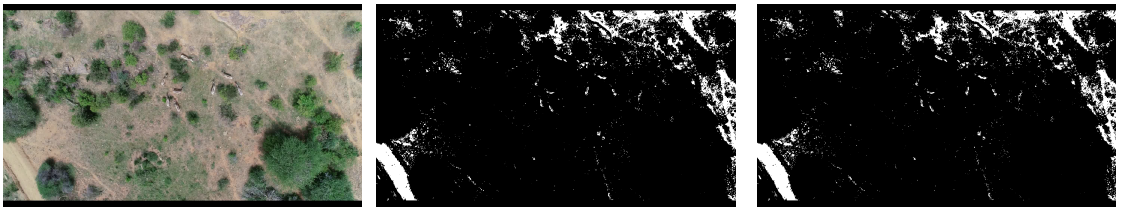
Furthermore this subsection groups the experiments, their results and interpretation under smaller parts introduced by headers. The headers are: the figures for the trackers' outputs for the video sequences, the description and interpretation of all outputs produced by the OpenCV trackers followed by the ones produced by the proposed trackers,



(a) (b) (c)

(d) (e) (f)

Figure 5.1: Segmentation examples for a frame (a) of the *cows video* and the sequence $ob002-01a$ when using as colour threshold either the mean value of a colour template (b) or the mean value of the green channel (c).



(a) (b) (c)

Figure 5.2: Examples of segmentation for a frame (a) of the sequence $ob090-03$ when it used as colour threshold either the mean value of a colour template (b) or the mean value of the green channel (c).

the introduction of new trackers and the presentation and interpretation of their results, the tables containing some statistics (SSIM values, frequencies for calling the second measurement provider, etc.) for some of the trackers, the role of the SSIM index, the frequencies associated with the particles' resampling for the trackers which involve the PF and the running time for the relevant trackers.

Video Sequences

The experiments were performed on four unlabelled video sequences from the MPI-AB dataset. These are sequences of lengths 4489, 4499, 4199 and 3595 frames. The first two sequences were taken from the video labelled *ob002-01*. Henceforth, they will be referred to as sequences $ob002-01a$ and $ob002-01b$, respectively. The two other sequences are

extracted from the videos *ob003-01* and *ob090-03*, respectively. As the data is obtained under an MOU, the study did not consider utilizing online labelling software to obtain the ground truth. Hand-labelling of data would have required a considerable amount of time.

Ground Truth and OpenCV Trackers

To label the video sequences, the eight open-source trackers implemented in OpenCV contribution version 4.1.0 were used. Each of them was run six times on each video sequence. The averaged results of the tracker (out of the eight) that visually performed the best has been chosen as ground truth. Therefore, all of the results for a given sequence are relative to the selected tracker. The mean of the eight trackers' predictions could have been used as ground truth. But it is not efficient to proceed like that, because some of the trackers lose the target. The trackers are Boosting [91; 129], Kernelized Correlation Filter (KCF) [102], Multiple Instance Learning (MIL) [94], Minimum Output Sum of Squared Error (MOSSE) [101], the Discriminative Correlation Filter with Channel and Spatial reliability (CSR-DCF) also referred to as the Spatial Channel and Spatial Reliability Tracking (CSRT) [103], Tracking-Learning-Detection (TLD) [95], Median Flow [89] and GOTURN [12].

All precision plots obtained on $ob002-01a$ are relative to MIL which was the best amongst the eight. Except for Boosting, all of the other trackers either confused the target with the background at some point or were unable to track the target to the end of this frame sequence. However, in most of the frames, both Boosting and MIL were tracking the target's shadow. Boosting started tracking the target's shadow from the 49^{th} frame while MIL started at the 121^{st} frame. Also, for most of the remaining frames, the target's centre estimate was closer to the target's actual centre for MIL than for Boosting. For sequences $ob002-01b$ and $ob003-01$, Boosting was retained while CSRT's outputs were used for the ground truth for the sequence $ob090-03$. In some of the frames, both CSRT and Boosting happened to track the target's back rather than its centre. This might not affect much the results with the actual ground truth, since the targets have small scales due to the drone's altitude. The difference between the animal centre and its back should only be of a few pixels. For the sequence $ob002-01a$ in particular, the precision plots would be translated by a distance in pixels close to the distance between the target and its shadow, relative to the precision plots with the actual ground truth.

Definition of Trackers which involve the PF

Next, for each video sequence, the main tracker is compared with two other trackers derived from the former and the seven remaining trackers of OpenCV using the precision plot. The main tracker, labelled *Tracker2*, is the one that relays between the colour segmentation and YOLOv3 triggered by the SSIM values. Note that in what follows results are also presented for two versions of *Tracker2* which are referred to as *Tracker2a* and *Tracker2b*. They differ in that for *Tracker*2*a*, YOLOv3 is called each time the SSIM value is below a set threshold while for *Tracker*2*b*, YOLOv3 is called not only based on the SSIM value but also if the current time step is even. This was performed to evaluate the influence on calling YOLOv3 periodically in an attempt to reduce *Tracker2*'s running time. The two other trackers (*Tracker1* and *Tracker3*) are the PF with the colour segmentation approach and the PF with YOLOv3 as measurement provider, respectively. They are evaluated to understand how the two measurement providers work independently compared to when they are relayed based on the SSIM index in *Tracker2*. The reader is invited to consult Table 5.1 for a summary of these trackers' components. The number of particles for all trackers involving the PF was set arbitrarily. For the sequences from the MPI-AB, a number of 200 particles is used, while for the *cows video*, the number of particles used is 2000.

Table 5.1: Components for *Tracker*1, 2*a*, 2*b*, 3 which are all based on the PF. CIS and NA are abbreviations for colour image segmentation and Non-Applicable, respectively.

Trackers	Measurement provider(s)	SSIM	YOLOv3's call
*Tracker*1	CIS	No	NA
*Tracker*2*a*	CIS and YOLOv3	Yes	If the SSIM index is below a set threshold.
*Tracker*2*b*	CIS and YOLOv3	Yes	If the SSIM index is below a set threshold and the time step k is even.
*Tracker*3	YOLOv3	No	For each time step k.

Initialisations of Trackers

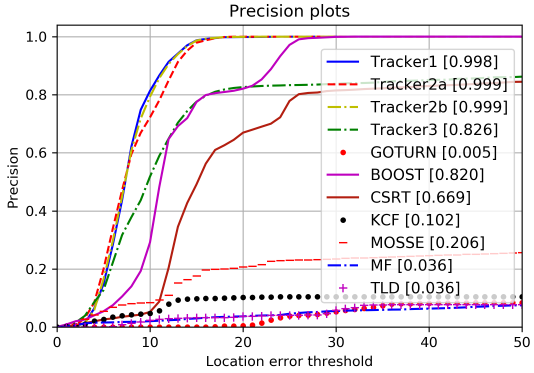
For each sequence, all of the eight trackers implemented in OpenCV were initialized with the same bounding box while *Tracker*1, 2 and 3 were initialized with the centre of this bounding box. The OpenCV trackers produce estimate of bounding boxes, the centres of which are considered as the trackers' estimates for the targets' locations. For all

of the sequences' frames, the bounding boxes are represented in blue, the trackers' positions estimates in green, YOLOv3's region of interest and the particles are represented in red. Note that for *Tracker*1, 2 and 3 (as well as new trackers which are introduced later) the bounding boxes are furnished by the measurement providers. For these methods, a threshold of value $2N/3$ was used for the step of particles resampling, following the work performed in [130]. As a reminder, the number of particles used is $N = 200$. This number was chosen arbitrarily.

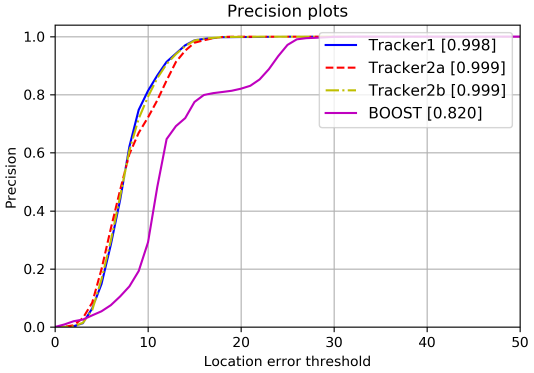
Figures for the Video Sequences' Results

All of the eleven trackers were executed six times on each sequence and the averaged results are considered instead of the results for single running, since some of the trackers do not produce the same track for different executions. In fact, some of the trackers like the PF-based trackers involve randomness. However, for the sequence $ob090-03$, GOTURN's results are not presented, because it was not possible to complete a single execution due to a lack of RAM on the laptop used. All results are shown in Figures 5.3, 5.4, 5.5 and 5.6 for sequences $ob002-01a$, $ob002-01b$, $ob003-01$ and $ob090-03$, respectively. For each of these figures, the first sub-figure shows all eleven trackers' precision plots with the ratio of frames which have the location error within 20 pixels. Indeed, the y-axis of the precision plot shows the ratio of frames where the prediction for the target location is less than a threshold for the location error (with respect to the ground truth). The x-axis presents different values of location error thresholds. For example, a value of 1 on the y-axis of a precision plot, indicates for a tracker that all of the locations predicted by this tracker for a video, are far from the actual locations by at most the corresponding location error threshold on the x-axis. In the legends, Median Flow is abbreviated as MF and Boosting as Boost. Note that these abbreviations are also used in tables presented in what follows. The second sub-figure shows the successful trackers, the third shows the latter's tracks and the fourth shows the failed tracks.

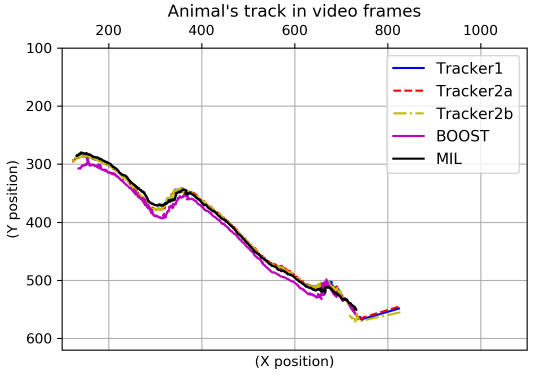
Regarding the intended application, a tracker is considered successful if it was able to keep tracking the target or the latter's salient features (like a shadow) in most frames from the beginning of a given sequence to its end. Across these sequences, *Tracker1*, Boosting, CSRT and MIL performed better than the other trackers including *Tracker2a*, *Tracker2b* and GOTURN, the only pure deep learning-based tracker considered.



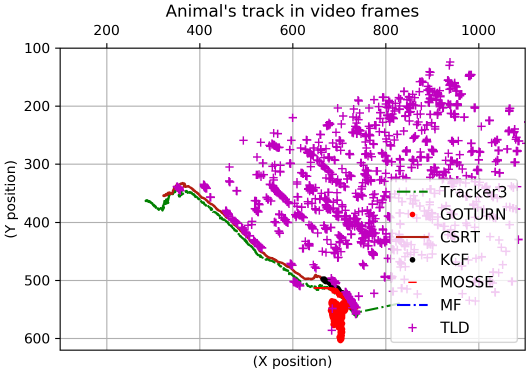
(a) Precision plots for all trackers.



(b) Precision plots for successful trackers.

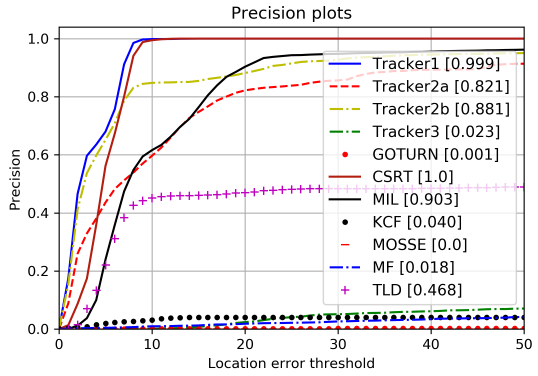


(c) Successful tracks.

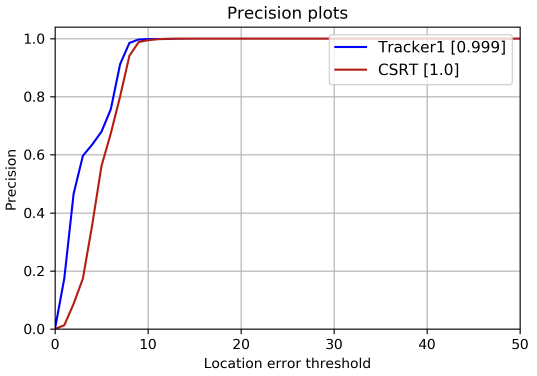


(d) Failed tracks.

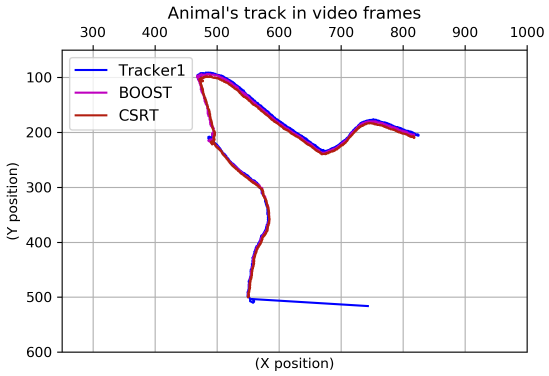
Figure 5.3: Results for the sequence $ob002 - 01a$. The ground truth is obtained from MIL. The tracking for $Tracker1, 2a, 2b$ and 3 start at a location different from the others, because the PF needs the first measurements to converge to the actual target's location. TLD's tracking is not consistent primarily due to TLD's detection module which scans the entire frame. CSRT and $Tracker3$ tracked the target for more than half the total frames.



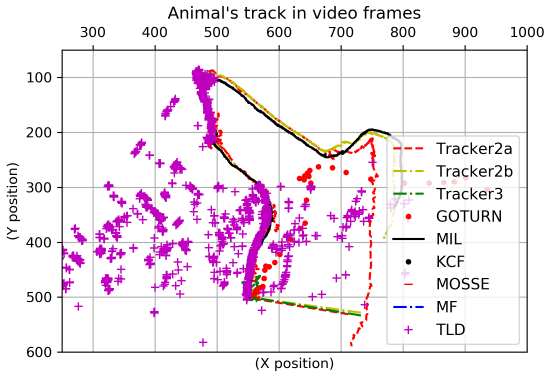
(a) Precision plots for all trackers.



(b) Precision plots for successful trackers.



(c) Successful tracks.



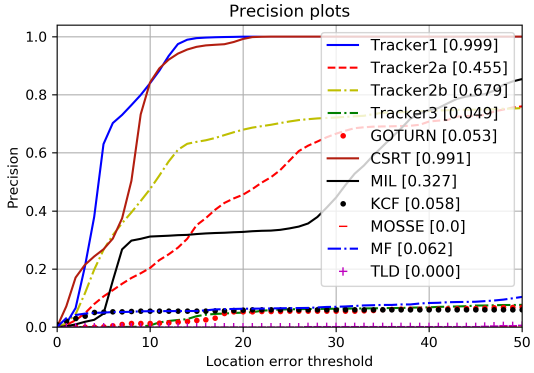
(d) Failed tracks.

Figure 5.4: Results for the sequence $ob002-01b$. The ground truth is obtained from Boosting. At the end of the sequence, *Tracker*2*a* shifted the target which stopped moving to a similar animal which was moving in a different direction. *Tracker*1 and CSRT seem to benefit from the white colour of the target.

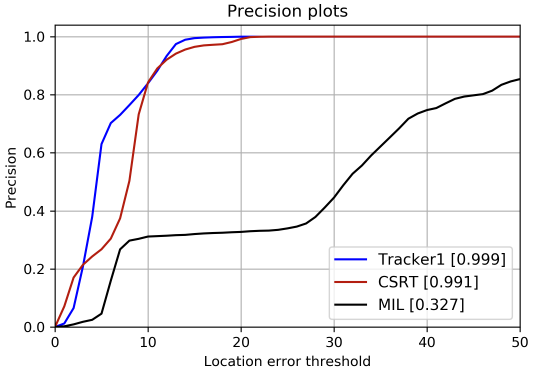
Description and Interpretation of the Results Produced by the OpenCV Trackers

To make clearer the explanations for the results produced by the OpenCV trackers, Table 5.2 provides summaries of the key features for these trackers. For more details, the reader is referred to Chapter 3.

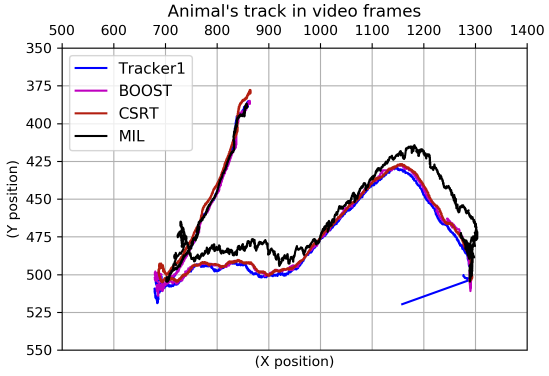
Boosting and MIL employ similar techniques, i.e. AdaBoost, for learning the target's appearances online. However, here, Boosting's tracking is more precise than MIL's tracking which tends to follow the target's shadow. This can be due to several reasons which happen successively. First, MIL drifts when the target's motion is relatively fast and when the target performs rotations that change its appearance. It was the case in the sequences $ob002-01a$ and $ob003-01$. MIL only tracks the target's location and does



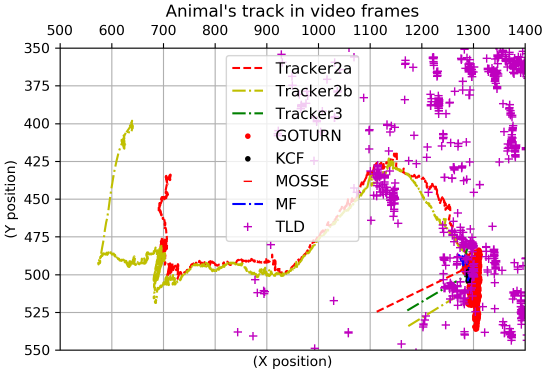
(a) Precision plots for all trackers.



(b) Precision plots for successful trackers.

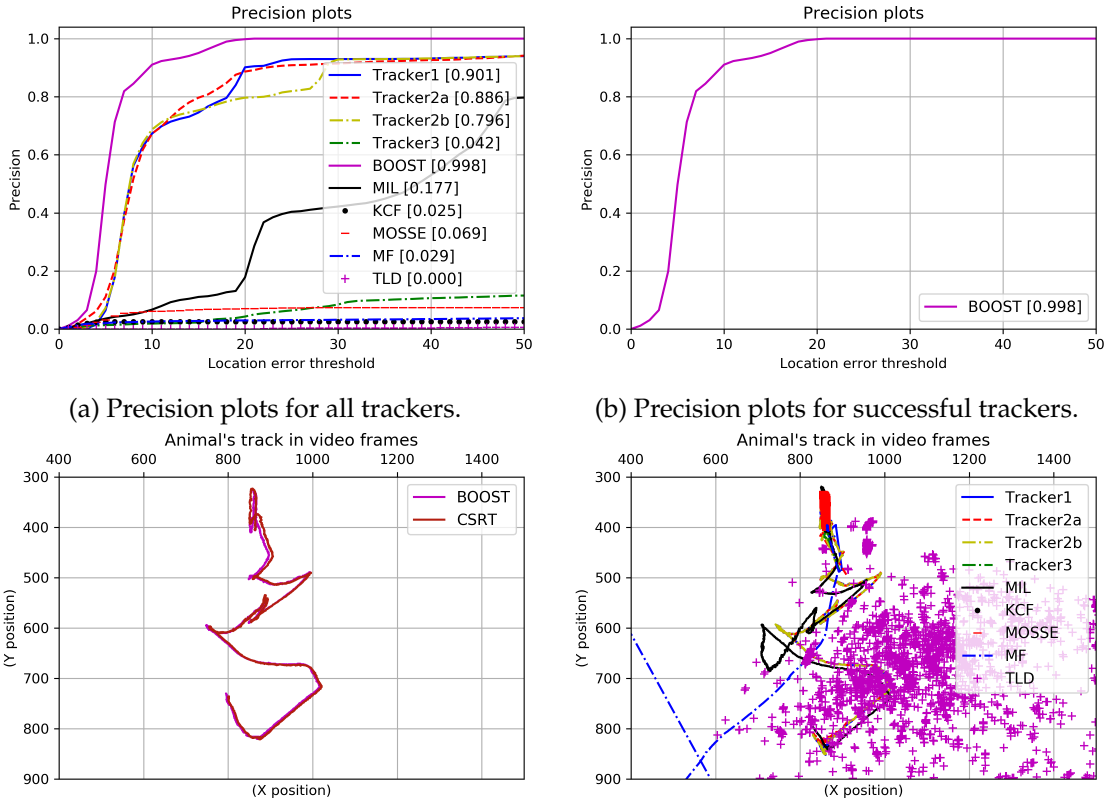


(c) Successful tracks.



(d) Failed tracks.

Figure 5.5: Results for the sequence $ob003-01$. The ground truth is obtained from Boosting. MIL lost the target momentarily. MIL managed to track the target due to the target's shadow. *Tracker*$2a, 2b$ were confused by the presence of two animals similar to the target. *Tracker*1's performance is due to the white colour of the target and the use of the anchor which helped in eliminating the two other animals present in the target's vicinity.



(a) Precision plots for all trackers.

(b) Precision plots for successful trackers.

(c) Successful tracks.

(d) Failed tracks.

Figure 5.6: Results for the sequence $ob090-03$. The ground truth is obtained from CSRT. *Tracker*1 lost the target in a few frames before the end of the sequence, because of the presence of an animal which resembles to the target, but it (the second animal) has a brighter colour. Similarly, *Tracker*$2a, 2b$ lost the target for the second animal. MIL started confusing the target with the similar animal earlier than *Tracker*$1, 2a$ and $2b$.

not consider modifications in the scale and the pose of the target for simplicity and to speed up tracking. Yet, MIL is expected to handle the target's moderate rotations, as well as changes in scale utilizing the Haar-like features [94]. The Haar-like features are obtained by computing the difference between the sum of pixels inside neighbouring rectangular regions at different locations of the relevant image patch. Therefore, they rely more on pixel intensities which do not vary under moderated changes in scale or orientation. Secondly, MIL relies on its previous estimated location to find the target in the current frame and to sample new learning instances. Therefore, if MIL has drifted at some point, it will hardly get back on track. Also, it is possible that the radius where the new samples are drawn is not adapted enough to the size of the tracked animal. Thirdly,

Table 5.2: Summary of the key features for OpenCV trackers. More details are available in Chapter 3. The table refers to concepts such as the correlation filter (CF), the kernelized correlation filter (KCF), the normalized cross-correlation (NCC) error, the discrete Fourier transform (DFT) and its inverse, the IDFT, as well as the fast Fourier transform (FFT) and its inverse, the IFFT.

Trackers	Description
Boosting	It uses Haar-like features, local binary patterns, orientation histograms, AdaBoost, weak and strong classifiers, selectors, and single images as learning instances.
MIL	It employs Haar-like features, AdaBoost, weak and strong classifiers and bags of images as learning instances.
CSRT	It relies on CFs to model the target and it involves both the FFT and the IFFT. It uses the channel and the spatial reliability maps, the HoG, and the colornames features.
KCF	It models the target with CFs, more specifically with KCF. It involves the DFT and the IDFT to accelerate computations and for circulant matrices for data augmentation.
MOSSE	It uses CF to model the target. It involves the FFT as well as its inverse, the IFFT to speed up calculations.
MF	It utilizes the target's keypoints, their median (of the keypoints), the Forward-Backward error, the NCC error, and the Lucas-Kanade tracker.
TLD	It exploits Median Flow for tracking, a detection module, and the P-N experts for learning the target's appearance.
GOTURN	It uses CNN kernels. It is based on two CNNs (to compare previous and current patches) and can track generic objects. It relies on regression.

the Haar-like features seem to easily accommodate dark and bright colours since as previously mentioned they are computed using pixels' intensities.

Boosting also uses Haar-like features and two other features which are the orientation histograms and the local binary patterns. It could be due to the Haar-like features that Boosting also tracked the target's shadow in the sequence $ob002-01a$. However, the use of the other types of features has probably contributed to the stability of the tracks produced by Boosting compared to the ones produced by MIL and by the other trackers, especially when the target is white or light grey against a green and/or a grey background as observed with the sequences $ob002-01b$ and $ob003-01$. For the sequence $ob090-03$ the target's colour is light brown but closer to the background's colour relative to the two other sequences. However, the target's back is whiter than the other parts. This supports the fact that Boosting's tracking is particularly reliable for targets that can be discriminated well against the background.

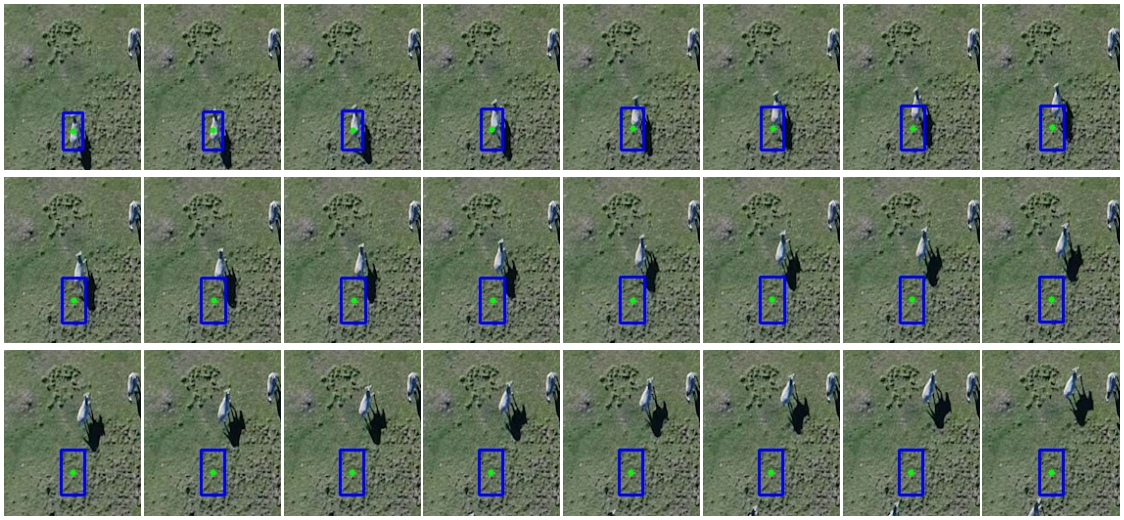
The poor performance observed with GOTURN on these video sequences might be

because GOTURN was not explicitly trained on the tracked animals and due to the scale of the animals. GOTURN is not always able to track generic objects despite its strategy of searching for the target in a region centred on the previous target location estimate, with dimensions twice the size of the previously-estimated bounding box [12]. Indeed, it was observed that for all of the sequences, the estimated bounding boxes tend to grow across frames up sometimes to reach an entire frame's size. It was the case for the sequence $ob090-03$. It seems GOTURN was not able to find the target in initial frames and ended up tracking the content of the bounding box due to its searching strategy. It explains why the RAM was not enough to complete the tracking of a single animal with GOTURN on this sequence. This highlights that with GOTURN, RAM would be likely to be a challenge in the field.

For all of the sequences, Median Flow was only able to track the very few first frames. To take one example, for the sequence $ob002-01b$ the tracker lost the target around the 120^{th} frame (Figure 5.7). It might either be because the target's motion was faster than what Median Flow can handle [89] or because it cannot find relevant keypoints to represent the targets through all frames. Similar reasons could partially explain TLD failure in tracking the actual targets in all sequences since Median Flow is employed as a tracker in TLD.

However, TLD has a detection module that corrects its tracker and works simultaneously. It should be the reason why TLD was able to track longer than Median Flow for the sequence $ob002-01b$ even if this detection module later induces the inconsistent tracking of the target (see the fourth sub-figure in Figure 5.4). The target changes its trajectory faster than Median Flow could follow, leaving more space in the bounding box to the background than necessary. TLD possesses the P-N experts. They provide training data to the detector. The detector also interacts with the P-N experts. Therefore, the false positive patch detected by the detector due to the tracking of the background by Median Flow (see the second last frame in the trajectories produced by TLD in Figure 5.7) might have compromised the P-N learning module. One could improve TLD by replacing the Median Flow tracker with a method less sensitive to fast motion and by reducing the detector's scanning area which is a whole frame to a smaller region, especially for applications where the presence of similar objects is expected.

Across all of the sequences, KCF only tracked the targets in initial frames. In sequences $ob002-01a$ and $ob003-01$, KCF stopped producing bounding box estimates as soon as the targets rotate while in $ob002-01b$ and $ob090-03$ it failed after the targets started moving. This could be because KCF can not handle changes in the object's orientation, relatively fast motion, and small-scale objects. MOSSE could not initialize the



(a) Median Flow

(b) TLD

Figure 5.7: Examples of tracks for Median Flow and TLD, respectively, for the first frames of the sequence $ob002 - 01b$, i.e. the first frame to the 346^{th} frame with an interval of 13 frames. They are displayed from left to right and top to bottom. Median Flow rapidly lost the target probably due to the speed and the scale of the target. TLD tracked the target longer than Median Flow, but it confused sometimes the target to the background (sub-figure in Figure 5.4) because its (TLD) detector started detecting the background as shown by the second last frame of the sequence for TLD.

tracking for sequences $ob002-01b$ and $ob003-01$. It might be because MOSSE could not capture the appearance of the type of animals present in these sequences using its correlation filters (CFs). As a matter of fact, the targets' appearances in the two videos are very similar. On the other hand, MOSSE initialized and kept in the first frames the tracking for sequences $ob002-01a$ and $ob090-03$. However, it started tracking the background in subsequent frames in the sequence $ob002-01a$ after the target changed its orientation and stopped the tracking in the sequence $ob090-03$ for the same reason. KCF and MOSSE behave similarly in tracking the targets. Since they are both CF-based object trackers, it is possible that such methods do not cope well with appearance modifications caused by abrupt changes in the target's pose.

This highlights some of the things that a good tracker should do: deal with changes in orientation/manoeuvrability, fast motion, and small scale. *Tracker*1 should be able to handle better these challenges if the target's colour can be well-discriminated relative to the background. Otherwise, there is a chance that false positives would be detected when these situations happen. Depending on how well YOLOv3 handles these situations, *Tracker*2, 3 could also be able to deal with such challenges. In addition, the presence of the PF in *Tracker*1, 2, 3 could help to mitigate the impact of these situations on the tracking.

For the sequence $ob002a$, similarly to MOSSE, CSRT started tracking the background after the target rotated although its tracking was longer. Since CSRT also relies on CF, both trackers' confusion between the background and the foreground might be due to the failure by the CF to adapt to rapid changes in the target's appearance. CSRT might have performed better than MOSSE and KCF by dint of the HoG and colorname features employed with concepts such as the spatial and the channel reliability maps. The CFs in CSRT seem to model the target's appearance quite well. Also, the CFs employed in CSRT seem more appropriate for the target's localisation compared to those utilised in MOSSE and KCF.

Description and Interpretation of the Results Produced by *Tracker*1, 2*a*, 2*b* and 3

The plots for the trajectories show that *Tracker*1, 2*a*, 2*b* and 3 seem not to start at the same position as other trackers, despite all being initialized to the same location as previously mentioned. This is because of the involvement of the PF. In the first frames of the sequences, the estimated locations for the targets are not accurate because the particles are initially randomly positioned. In order for the PF to quickly converge to locations which are near to the actual targets' positions, for all the sequences, the anchor in the first sixty frames was set to be the measurements obtained at the time step $k-1$. For

further frames, there is a shift to the previous PF's estimates as presented in the previous chapter. It implies that if the initially provided measurements are not reliable enough the tracking can fail. Yet, to mitigate this, one could rather concentrate on the prior. The fact that the particles are initially dispersed in the whole initial frames, partially explains why *Tracker*3 fails in most sequences compared to *Tracker*1 and *Tracker*2. Another related reason is that of the absence of a second measurement provider for *Tracker*3.

For *Tracker*2, this situation is primarily avoided by the presence of the colour image segmentation approach. In fact, the latter could have provided good measurements to the PF in such a way that YOLOv3 would have not even been involved in detection and tracking in these frames. It is even more apparent when observing *Tracker*2*b*'s performance relative to *Tracker*2*a*. Indeed, since for *Tracker*2*b*, YOLOv3 only furnishes measurements when the SSIM index is below the set threshold and when the current time step is even, the colour segmentation produces more measurements than YOLO. Yet, *Tracker2* can also later lose the target due to the colour image segmentation.

YOLOv3 misses a significant number of detections. Indeed, As an example for *Tracker*2*a* regarding the sequence $ob003 - 01$, the SSIM index was below the set threshold in 3354 frames out of 4199 frames (see in the second column, the third and the fourth rows of Table 5.8). However, YOLOv3 was only able to perform detections in 299 frames. Similar results are presented for the other sequences in Tables 5.4, 5.6 and 5.10. This significant number of missed-detections by YOLOv3 can be justified for several reasons.

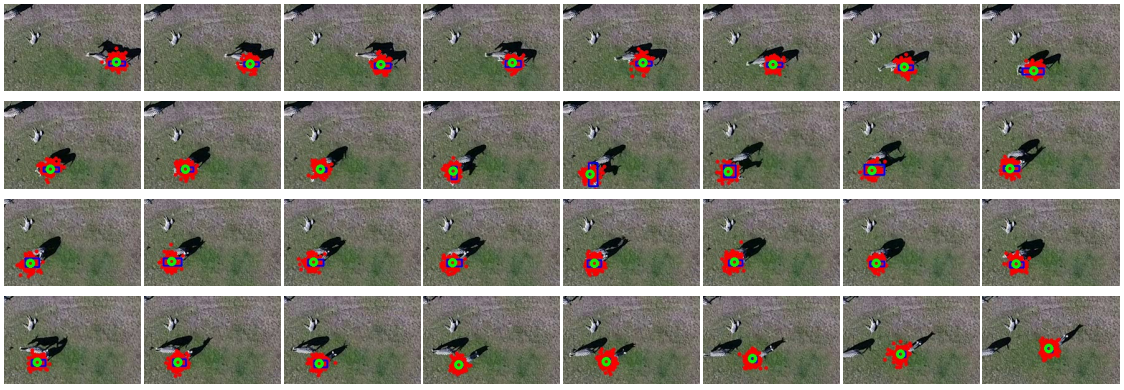
- Despite using YOLOv3, the network still struggles with small object detections.
- The network was not explicitly trained on the target animal for its accurate detection. Therefore, in the presence of several similar animals, it can not differentiate between them. Yet, even if YOLOv3 was trained on the specific targets, it is possible that the network will not be able to accurately discriminate the animals since they look very alike from high altitudes. One might consider training YOLOv3 on the species of a group of animals viewed from above instead of on a particular animal in this group, especially if one is more interested in tracking a group of animals than a specific animal.
- The ROI is involved in the proposed VOT solution, for the reduction of the YOLOv3's search region. But the ROI is too small to enable YOLOv3 to detect animals in the corresponding small image patch. However, it is important to use the ROI and moreover to tune its dimensions to the target's size when using YOLOv3 in this tracking framework because it helps YOLOv3 to focus on the target and to considerably reduce YOLOv3's running time. The reduction of YOLOv3's running time

is beneficial for the intended real-time application. In other words, increasing the ROI's dimensions results in increasing the missed detections (in presence of other similar animals) and YOLOv3's running time.

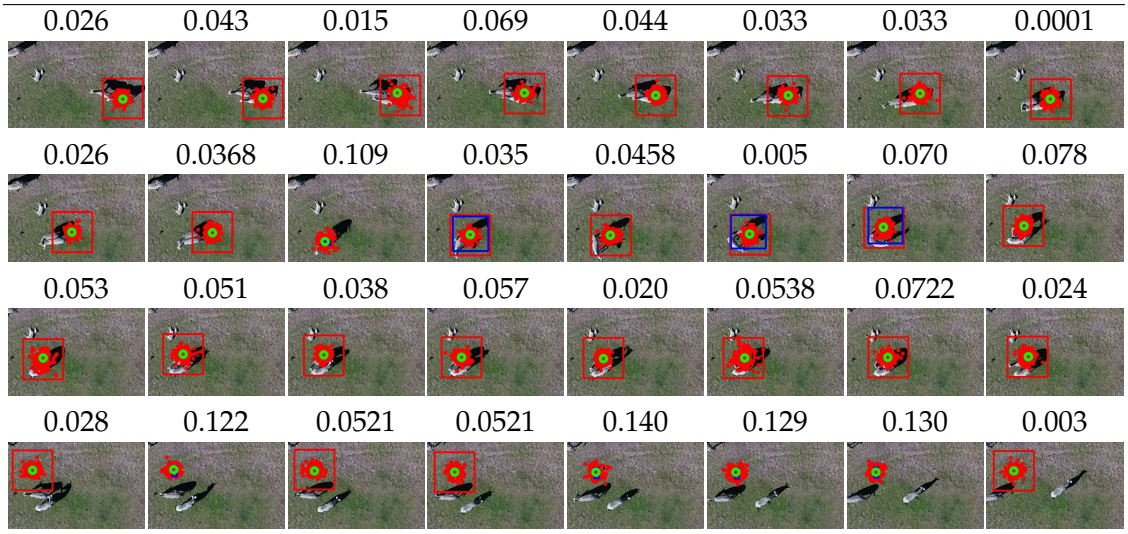
Figure 5.8 and 5.9 show the outputs for single runs of *Tracker*1, 2*a*, Boosting, CSRT and MIL on a challenging tracking situation of the sequence $ob003-01$. Averaged results are not considered here because one needs to refer to ROIs and bounding boxes to qualitatively interpret *Tracker*1 and 2*a*'s outcomes (although the bounding boxes are furnished by the measurement providers for *Tracker*1 and 2*a*). For the averaged results, at a time step k, the bounding boxes and the ROIs do not always contain the target even though the estimated location lies on the target. It is because, depending on the measurement furnisher, the proposed tracking framework does not always produce the same results since it relies on the anchor which has some uncertainty.

The challenging situation of the sequence $ob003-01$ is that of an animal similar to the target which comes into the vicinity of the target and takes an opposite direction after both met. In addition, there is a third animal that resembles the two other animals. It is resting in the same area. *Tracker*2*a* follows one of the three animals because YOLOv3 is unable to distinguish the target from the two other animals. In some of the frames of Figure 5.8, one can see that the ROI (in red) contains the target and the other moving animal. YOLOv3 randomly detected the other animal (see bounding boxes in blue). The SSIM index kept on switching both measurement providers due to the difficulty of distinguishing both animals since they have small scales. Next, the colour image segmentation detected the third animal due to its brightness and the presence of the anchor. *Tracker*2*a* got stuck tracking the third animal. In this case, the anchor seems not to be helpful. However, it is beneficial when reliable measurements are furnished, i.e when these measurements are sampled near the target. It is by dint of the anchor and the foreground's colour that *Tracker*1 successfully handled the sequence $ob003-01$'s challenging situation. The anchor largely contributed to eliminating the second animal which was farther from it than from the actual target. Boosting and CSRT were also able to successfully track the target. MIL lost the target momentarily as it was tracking its shadow but, MIL found the target in subsequent frames.

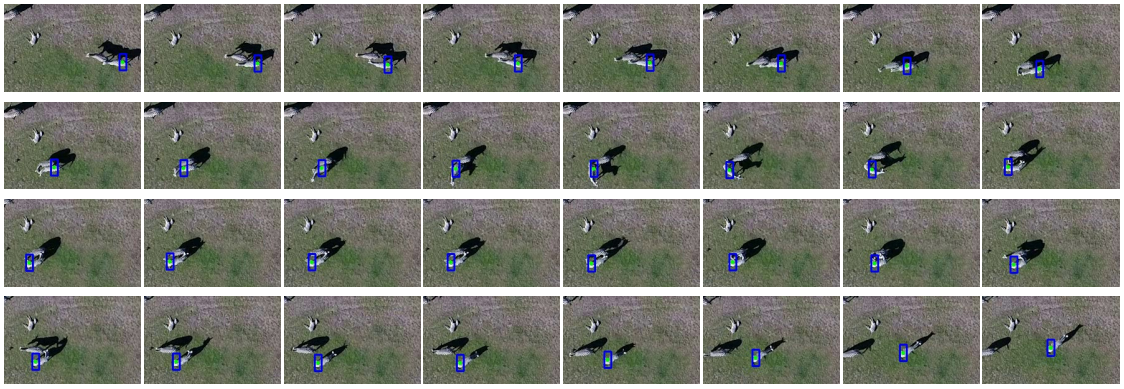
It is clear that the situation becomes more difficult for the trackers as altitude increases/scale decreases, motion increases, etc. All of the trackers fail in these cases. Indeed, some tests were performed on other sequences from videos of the MPI-AB dataset, such as the sequence $ob029-01$ which involves more challenging tracking situations. The case where both foreground and background have similar colours was challenging for the colour image segmentation which performed well overall for the four sequences.



(a) *Tracker*1

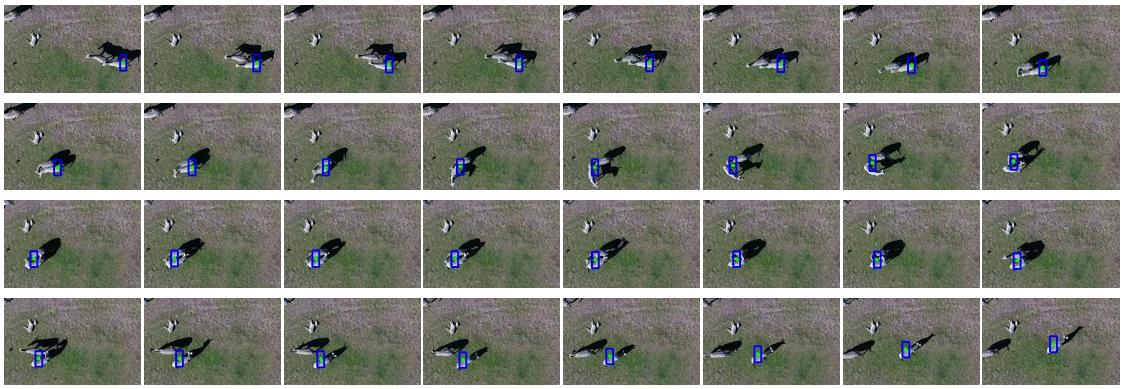


(b) *Tracker*2*a*

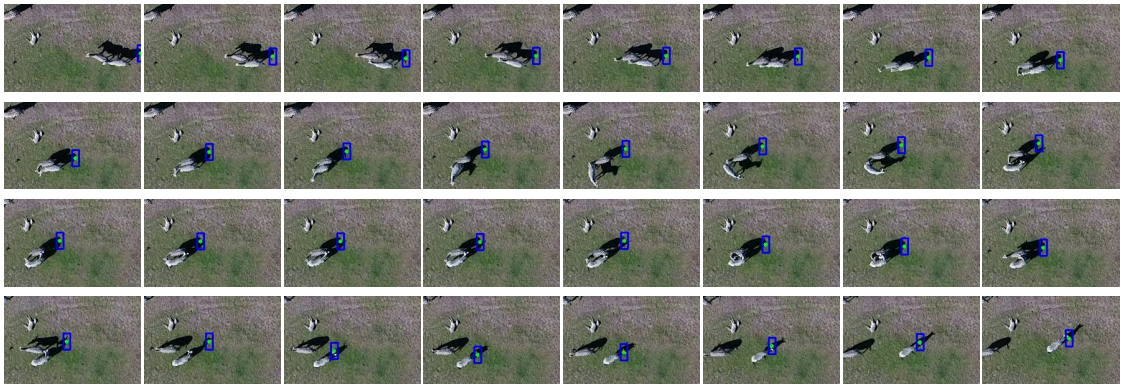


(c) Boosting

Figure 5.8: Examples of the performance of *Tracker*1, 2*a* and Boosting, respectively, for single runs on the sequence $ob003 - 01$. Frames are displayed from left to right and top to bottom with interval 49 frames, starting from the 1750^{th} frame. For *Tracker*2*a*'s sequence output, the relevant SSIM value is displayed above each frame. Boosting and *Tracker*1 produced reliable trajectories. *Tracker*1 benefited from the white colour of the target and the involvement of the anchor. For *Tracker*2*a*, YOLOv3 missed several detections primarily due to the target's scale.



(a) CSRT



(b) MIL

Figure 5.9: Examples of the performance of CSRT and MIL, respectively, for single runs on the sequence $ob003 - 01$. Frames are displayed from left to right and top to bottom with interval 49 frames, starting from the 1750^{th} frame. MIL tracked the target's shadow sometimes and lost the target momentarily.

Introduction to New Trackers, Description and Interpretation of their Results

Following the observation that YOLOv3 misses a significant number of detections and the hypothesis that the proposed tracking framework will work better if the second measurement provider furnishes accurate measurements, more tests were carried out with the introduction of new trackers. They are addressed as *Tracker*4, 5, 6 and are basically *Tracker*2*a* with as second measurement provider either Boosting, CSRT, or MIL, respectively, to replace YOLOv3. Table 5.3 summarizes these trackers' components. However, they do not involve the ROI. Indeed, some tests were performed with the ROI, and it was noticed that *Tracker*4, 5, 6 did not perform well in presence of the ROI. For the sake of brevity, these results are not presented here. The results for *Tracker*6 are also not presented because they were inconsistent, in particular for the sequences $ob002 - 01a$ and $ob090 - 03$ for which more tests were performed in an attempt to assess *Tracker*6's performance.

It was observed that *Tracker*6 presented a non-negligible number of outliers. For 20 tests on these two sequences, *Tracker*6 lost the target for about 30% of the tests. It should be because MIL's tracking varies more from one execution to another compared to Boosting and CSRT. In addition, the PF relies on the motion model which involves noise and the colour image segmentation which produces several false negatives. This suggests that the proposed tracking framework can sometimes produce outliers. The combination of the three elements i.e. the noise in the PF's motion model, the colour image segmentation and MIL results in the instability of *Tracker*6.

Table 5.3: Components for *Tracker*4, 5, 6 which are all based on the PF and do not involve ROI. CIS is the abbreviation for colour image segmentation. The second measurement provider is called when the current SSIM value is below the set threshold.

Trackers	Measurement provider(s)	SSIM
*Tracker*4	CIS and Boosting	Yes
*Tracker*5	CIS and CSRT	Yes
*Tracker*6	CIS and MIL	Yes

The ROI is more relevant for a deep learning detection method than a machine learning and/or a CF-based tracker. It is because Boosting and CSRT do not necessarily find the target at the expected location since the ROI constantly changes as it is centered at the anchor. In addition, the size of the animals degrades the location accuracy for Boosting and CSRT. In presence of the ROI, these trackers tend to learn negative patches or false positives when it is their turn to furnish the measurements. Therefore, these meth-

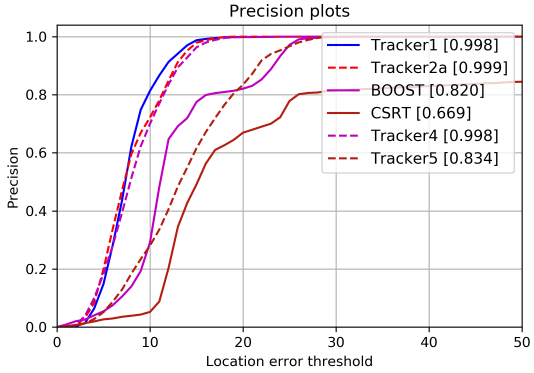
ods are sensitive to the precision in the expected locations for the target. This situation is essentially a missing data (missing frames) problem.

Other results which also provide evidence to the importance of the precision in the target's expected location for Boosting and CSRT (especially for small targets) are presented in Figures 5.10, 5.11, 5.12 and 5.13. Each figure corresponds to a given sequence. The first sub-figure and the one below it are the precision plots and the corresponding tracks for *Tracker*4, 5 when the colour image segmentation is relayed by Boosting or CSRT without having the latter learning the target's appearances during the former's turns. Boosting and CSRT are deliberately prevented from tracking the target when the colour image segmentation is providing the actual measurement, in order to test the robustness of Boosting and CSRT to missing data and hence the impact on *Tracker*4, 5. This case resembles the problem of the precision in the target's location with the ROI since it is a missing data situation.

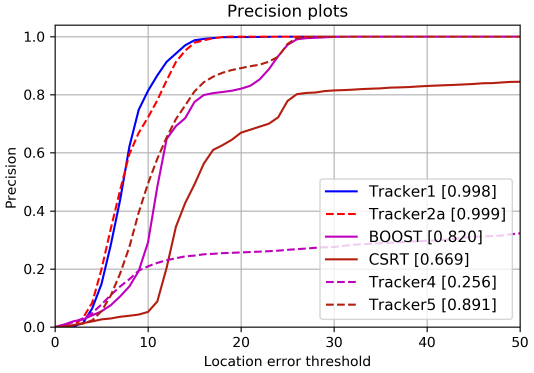
CSRT has been affected by the situation of missing data more than Boosting. Compared to Boosting, CSRT behaved as YOLOv3 by missing 109 and 1649 detections on average for the sequences $ob002 - 01a$ and $ob090 - 03$, respectively, when it could not learn the target's appearances during the colour image segmentation's turns with *Tracker*5. This performance is worse than that for Boosting and CSRT alone, as well as for the cases where they (Boosting and CSRT) are integrated into the proposed framework and they cannot continue tracking during the colour image segmentation's turns (see the top right sub-figure for the precision plots and the sub-figure below it for the track's plots for Figures 5.10, 5.11, 5.12 and 5.13). For all the sub-figures, the precision plots of Boosting and CSRT, *Tracker*1 and 2*a* are included for comparison.

In other words, *Tracker*4 and 5 perform better or similarly in general compared to the other trackers, when Boosting and CSRT can track independently of the colour image segmentation but can only provide measurements when the SSIM index decides it. It is more like a fusion of tracking from the colour image segmentation and either Boosting or CSRT. The plots for the situation of missing data suggest that *Tracker*4 and 5 can handle wrong and/or missing data sometimes. Indeed, for the sequence $ob002 - 01a$, *Tracker*4, 5 produced reliable trajectories despite the fact that Boosting was detecting sometimes the background towards the end of the sequence while as mentioned before *Tracker*5 missed detections in some of the frames where it was called. It is due to the use of the last measurement when no new measurement is available at time step k and to the utilization of the PF which takes few time steps to converge to a given location.

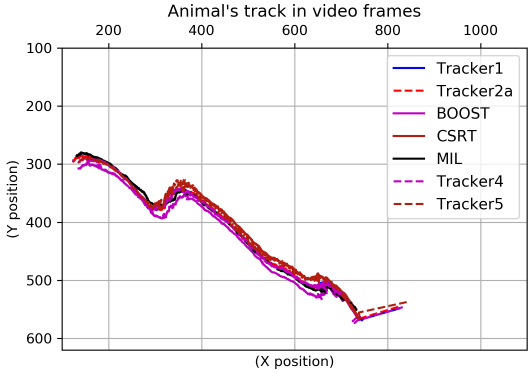
In general, by taking a close look at the plots corresponding to the case where the second measurement furnisher was tracking in parallel, i.e. when there is no missing



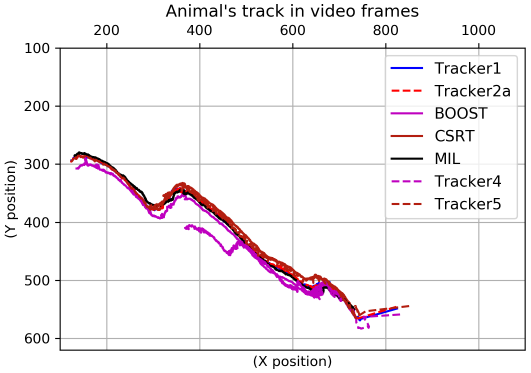
(a) Precision plots with missing data.



(b) Precision plots without missing data.

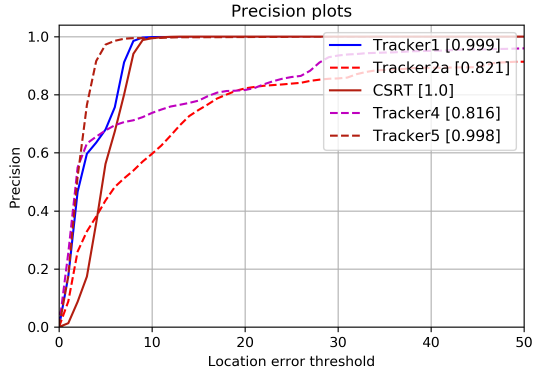


(c) Trajectories for missing data.

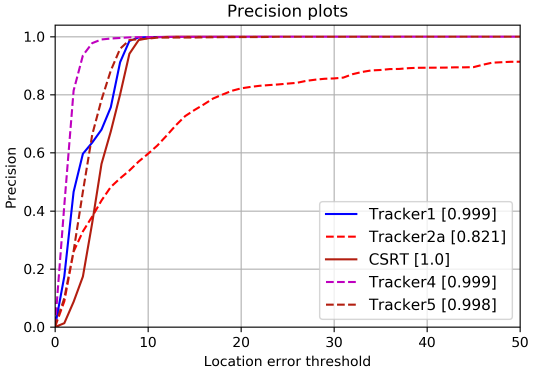


(d) Trajectories without missing data.

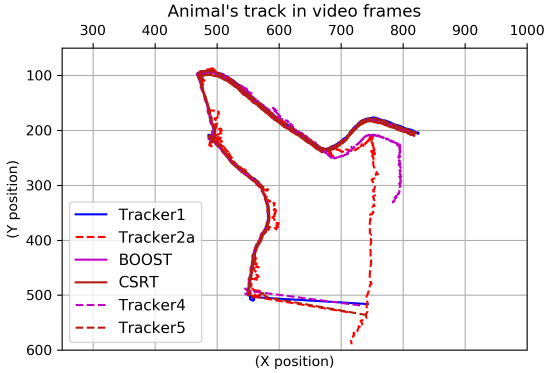
Figure 5.10: *Tracker*4, 5's results for the sequence $ob002 - 01a$ when the corresponding second measurement provider (Boosting, CSRT) is either missing data or not. For Sub-figures (a) and (c), Boosting and CSRT are deliberately prevented from learning the target's appearance during the turns of the colour image segmentation to assess the performance of *Tracker*4, 5 when Boosting and CSRT furnish unreliable measurements sometimes. CSRT missed 109 frames on average in this case. The performance of *Tracker*1, 2*a*, Boosting, CSRT, and the trajectories they produced are recalled for comparison with the ones for *Tracker*4, 5. The ground truth is obtained from MIL. For Sub-figures (b) and (d), *Tracker*5 performed better than CSRT.



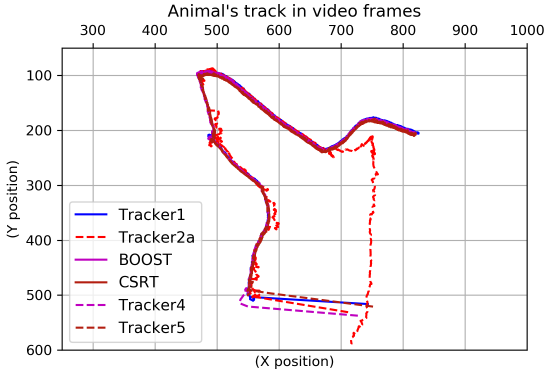
(a) Precision plots with missing data.



(b) Precision plots without missing data.

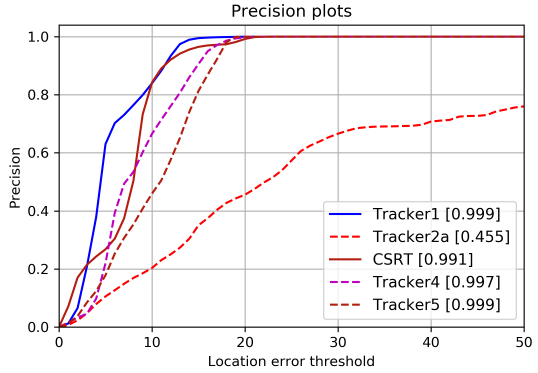


(c) Trajectories for missing data.

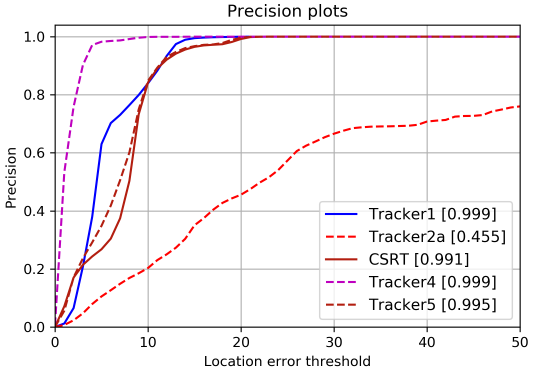


(d) Trajectories without missing data.

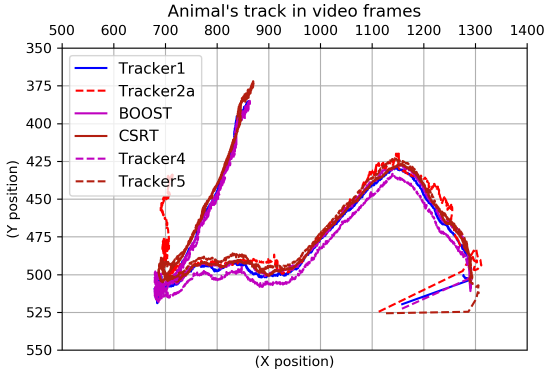
Figure 5.11: *Tracker*4, 5's results for the sequence $ob002 - 01b$ when the corresponding second measurement provider (Boosting, CSRT) is either missing data or not. For Sub-figures (a) and (c), Boosting and CSRT are deliberately prevented from learning the target's appearance during the turns of the colour image segmentation to assess the performance of *Tracker*4, 5 when Boosting and CSRT furnish unreliable measurements sometimes. The performance of *Tracker*1, 2*a*, CSRT, and the trajectories they produced are recalled for comparison with the ones for *Tracker*4, 5. The ground truth is obtained from Boosting. For Sub-figures (b) and (d), *Tracker*4 outperforms the other trackers while *Tracker*5 performed slightly better than CSRT.



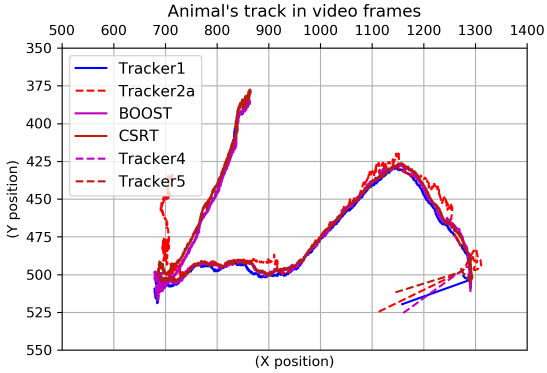
(a) Precision plots with missing data.



(b) Precision plots without missing data.

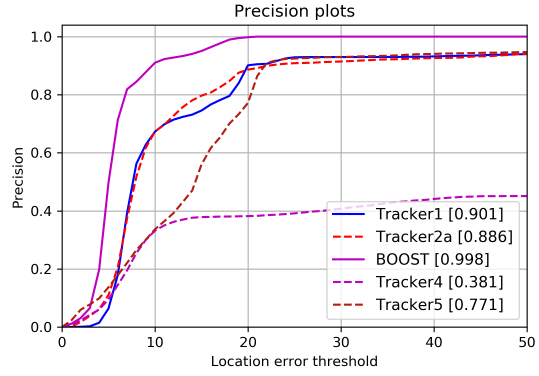


(c) Trajectories for missing data.

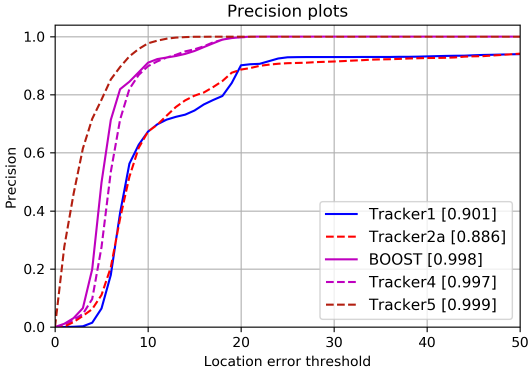


(d) Trajectories without missing data.

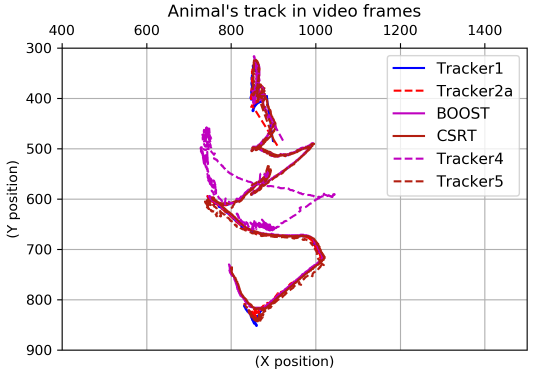
Figure 5.12: *Tracker*4, 5's results for the sequence $ob003 - 01$ when the corresponding second measurement provider (Boosting, CSRT) is either missing data or not. For Sub-figures (a) and (c), Boosting and CSRT are deliberately prevented from learning the target's appearance during the turns of the colour image segmentation to assess the performance of *Tracker*4, 5 when Boosting and CSRT furnish unreliable measurements sometimes. The performance of *Tracker*1, 2*a*, CSRT, and the trajectories they produced are recalled for comparison with the ones for *Tracker*4, 5. The ground truth is obtained from Boosting. For Sub-figures (b) and (d), *Tracker*4 outperforms the other trackers while *Tracker*5 performed better than CSRT.



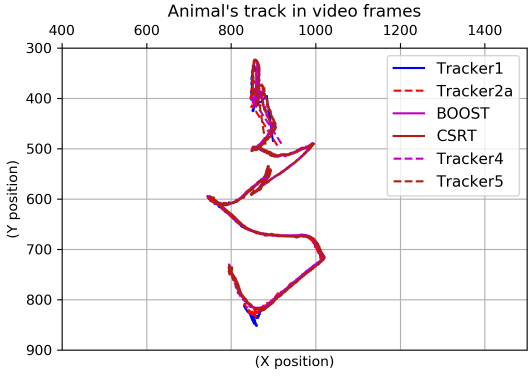
(a) Precision plots with missing data.



(b) Precision plots without missing data.



(c) Trajectories for missing data.



(d) Trajectories without missing data.

Figure 5.13: *Tracker*4, 5's results for the sequence *ob*090 − 03 when the corresponding second measurement provider (Boosting, CSRT) is either missing data or not. For Sub-figures (a) and (c), Boosting and CSRT are deliberately prevented from learning the target's appearance during the turns of the colour image segmentation to assess the performance of *Tracker*4, 5 when Boosting and CSRT furnish unreliable measurements sometimes. CSRT missed on average 1649 detections in this case. The performance of *Tracker*1, 2*a*, CSRT, and the trajectories they produced are recalled for comparison with the ones for *Tracker*4, 5. The ground truth is obtained from CSRT. For Sub-figures (b) and (d), *Tracker*5 outperforms the other trackers while *Tracker*4 performed similarly to Boosting. *Tracker*1, 2*a* shifted to a similar animal in the target's vicinity, in a few frames before the end of the sequence.

data (for Boosting and CSRT), *Tracker*4 and 5 performed better than the other trackers on the sequences where their second measurement provider was used to get the ground truth. Such behaviour is expected since some of these trackers' estimates coincide with the ground truth. It shows that the proposed tracking framework works quite well when the second provider produces accurate measurements. It demonstrates the suitability of a precise method as the second measurement furnisher in the proposed tracking framework (even if it might be at the cost of running-time). On the other hand, *Tracker*5 performed better than CSRT alone in all of the video sequences (where CSRT is not the reference-tracker), because it also involves the colour image segmentation. The performance observed with both *Tracker*4 and 5 validates the hypothesis about the need for a method able to accurately identify the target as the second measurement provider.

In the case where the second provider is accurate and independent of the colour image segmentation, the latter can rely on the former. Indeed, such a second measurement provider can catch up with the errors of the colour image segmentation. If the second measurement provider interacts with the colour image segmentation by the intermediary of the anchor (as is the case with YOLOv3 in *Tracker*2), both measurement providers can work in a complementary way and correct each other. However, they can also induce failures in each other. In all of the cases (i.e. if the second measurement provider tracks independently or dependently of the colour image segmentation), when both measurement providers are prone to persistent errors, they can induce failures in the proposed tracking framework and they can lead therefore to the inability of the latter to recover.

Tables with Statistics (about SSIM, Running Times and Others) for some of the Trackers

Tables 5.4, 5.6, 5.8 and 5.10 show results for trackers which are based on the proposed tracking framework, for the sequences $ob002-01a$, $ob002-01b$, $ob003-01$ and $ob090-03$, respectively. In these tables, NS, NDS, RT, PFR, and NR are the number of frames where the SSIM values were below the set threshold, the number of actual detections in these frames, the total running time, the running time per frame, and the number of times the particles have been resampled, respectively. For the trackers which have as second measurement provider one of the OpenCV trackers, the OpenCV tracker's name is given in brackets as a reminder. Note that the range of values for statistics about the SSIM index comprises between 0 and 1 instead of -1 and 1. It is because the absolute value of the SSIM index is considered.

SSIM Index values

The range of values for the SSIM index for the video sequences is between 0 and 0.3. One might think that the maximal value of the SSIM index should be very close to 1. since the initial target's template has to match the initial animal's appearance. However, it is not the case here because of the precision errors that occurred when determining the initial location. Indeed, to find the initial location for the target and the initial template, The GNU image manipulation program (GIMP) was utilized. The imprecision in reading the target's location influenced the value of the SSIM in the first frame. For each sequence, the previous statement was verified by sampling multiple patches around the initial location and by comparing each of them to the initial template. The values were either 1. or very close to 1. as expected.

This test confirms that the SSIM index might not be appropriate to handle efficiently change detections in the target's appearance when the target has a small scale as is the case for drones flying at high altitudes. However, the SSIM index is still a good basis for relaying two measurement providers for small-scale targets.

Frequency of Particle Resampling

Tables 5.4, 5.6, 5.8 and 5.10) also report the number of times the particles were resampled. For all of the sequences, resampling happened at almost every frame. Two reasons which can explain this situation are identified. The resampling threshold of $2N/3$ is too high or the majority of the particles' weights at a given time step have small values. However, the latter hypothesis can be eliminated because the qualitative results show that the particles are concentrated around the PF estimates in most frames. Also, if one considers the frequency of the resampling itself, it means that the particles are almost always equally likely. Therefore, it should be that the resampling threshold is not small enough.

To verify the previous hypothesis, tests were performed for all of the trackers involving the PF with a threshold of $2N/60$, apart from the ones where Boosting and CSRT do not learn independently of the colour image segmentation (as these cases are no longer relevant since it was established that preventing them from learning the target's appearance during the turns of the colour image segmentation is not beneficial for *Tracker*4 and 5). It was observed that with such a threshold, the resampling happens at least 7 times fewer and at most 6 times fewer than what was observed with $2N/3$ for all the sequences with similar performance.

Running Times for some of the Trackers

Regarding the OpenCV trackers, TLD has the longest running time but TLD was not able to consistently track the targets. MIL was slower than Boosting. It is probably because MIL trains all of the weak classifiers on all the instances in a bag while Boosting only trains on one sample.

For all of the sequences, *Tracker*2*b*'s running-time is lower than *Tracker*2*a*'s one as expected. Recall that in *Tracker*2*b*, YOLOv3 is called not only when the SSIM index is below the set threshold but also when the time step k is even. In *Tracker*2*a*, YOLOv3 is roughly called twice the number of times it is called in *Tracker*2*b*. The latter's performance did not degrade compared to the former due to the colour segmentation as previously mentioned.

Since the colour segmentation is used in *Tacker*4, 5, these trackers' running-times are compromises between *Tracker*1's running-time and the one of either Boosting or CSRT. *Tracker*4 and *Tracker*5 can process 28.7 and 20.5 fps at most, respectively here. The proposed tracking framework is suitable for real-time tracking as shown by the running times of *Tracker*4, 5 when the second measurement provider is also fast. They benefit from the speeds of the colour image segmentation, Boosting, and CSRT.

Table 5.4: Data for the trackers based on the proposed tracking framework for the sequence $ob002 - 01a$. The latter sequence has 4489 frames in total. The SSIM threshold here is 0.1. SSIM index statistics are between 0 and 1, given that the absolute value of the SSIM index is considered here. NS, NDS, RT, PFR, and NR are the number of frames where the SSIM values were below the set threshold, the number of actual detections in these frames, the total running time, the running time per frame, and the number of times the particles have been resampled, respectively.

	Tracker			
	2a	2b	4 (Boost)	5(CSRT)
NS (frames)	2392	1109	1845	2391
NDS (frames)	633	386	1845	2391
SSIM Mean	0.099	0.103	0.106	0.100
SSIM STD	0.027	0.027	0.013	0.024
SSIM Max	0.199	0.202	0.180	0.272
SSIM Min	0.031	0.028	0.05	0.03
RT (min)	5.385	4.524	4.450	4.196
PFR (fps)	13.9	16.5	16.8	17.8
NR	4489	4487	4049	4347

Table 5.5: Per frame and total running times for *Tracker*1, 3 and some of the OpenCV trackers for the sequence $ob002 - 01a$. RT and PFR are the total running-time and the running time per frame, respectively.

	Tracker		Boost	CSRT	MIL
	1	3			
RT (min)	3.740	6.532	1.983	2.200	6.087
PFR(fps)	20	11.5	37.7	34	12.3

Table 5.6: Data for the trackers based on the proposed tracking framework for the sequence $ob002 - 01b$. The latter sequence has 4499 frames in total. The SSIM threshold here is 0.1. SSIM index statistics are between 0 and 1, given that the absolute value of the SSIM index is considered here. NS, NDS, RT, PFR, and NR are the number of frames where the SSIM values were below the set threshold, the number of actual detections in these frames, the total running time, the running time per frame, and the number of times the particles have been resampled, respectively.

	Tracker			
	2a	2b	4 (Boost)	5(CSRT)
NS (frames)	2793	1033	2115	3155
NDS (frames)	411	240	2115	3155
SSIM Mean	0.085	0.105	0.105	0.076
SSIM STD	0.043	0.043	0.044	0.040
SSIM Max	0.242	0.307	0.294	0.270
SSIM Min	0.0002	0.005	0.003	0.0003
RT (min)	3.33	1.956	2.810	3.656
PFR (fps)	22.5	38.33	26.7	20.5
NR	4495	4497	4497	4495

Table 5.7: Per frame and total running times for *Tracker*1, 3 and some of the OpenCV trackers for the sequence $ob002 - 01b$. RT and PFR are the total running-time and the running time per frame, respectively.

	Tracker		Boost	CSRT	MIL
	1	3			
RT (min)	1.042	4.233	1.965	3.031	6.285
PFR(fps)	72	17.7	38.2	24.7	11.9

Table 5.8: Data for the trackers based on the proposed tracking framework for the sequence $ob003 - 01$. The latter sequence has 4199 frames in total. The SSIM threshold here is 0.08. SSIM index statistics are between 0 and 1, given that the absolute value of the SSIM index is considered here. NS, NDS, RT, PFR, and NR are the number of frames where the SSIM values were below the set threshold, the number of actual detections in these frames, the total running time, the running time per frame, and the number of times the particles have been resampled, respectively.

	Tracker			
	2a	2b	4 (Boost)	5(CSRT)
NS (frames)	3354	1820	3243	3681
NDS (frames)	299	307	3243	3681
SSIM Mean	0.051	0.045	0.059	0.045
SSIM STD	0.027	0.026	0.029	0.025
SSIM Max	0.197	0.211	0.253	0.230
SSIM Min	0.002	0.001	0.002	0.0004
RT (min)	3.693	2.802	2.436	3.921
PFR (fps)	19	25	28.7	17.8
NR	4186	4187	4199	4198

Table 5.9: Per frame and total running times for *Tracker*1, 3 and some of the OpenCV trackers for the sequence $ob003 - 01$. RT and PFR are the total running-time and the running time per frame, respectively.

	Tracker		Boost	CSRT	MIL
	1	3			
RT (min)	1.461	3.982	2.436	3.129	5.852
PFR(fps)	47.9	17.6	28.7	22.4	12

Table 5.10: Data for the trackers based on the proposed tracking framework for the sequence $ob090-03$. The latter sequence has 3595 frames in total. The SSIM threshold here is 0.1. SSIM index statistics are between 0 and 1, given that the absolute value of the SSIM index is considered here. NS, NDS, RT, PFR, and NR are the number of frames where the SSIM values were below the set threshold, the number of actual detections in these frames, the total running time, the running time per frame, and the number of times the particles have been resampled, respectively.

	Tracker			
	2a	2b	4 (Boost)	5(CSRT)
NS (frames)	2293	1083	2464	2278
NDS (frames)	25	18	2463	2278
SSIM Mean	0.095	0.099	0.092	0.091
SSIM STD	0.036	0.035	0.038	0.036
SSIM Max	0.242	0.253	0.242	0.24
SSIM Min	0.	0.010	0.01	0.002
RT (min)	3.06	2.497	2.570	2.980
PFR (fps)	19.6	24	23.3	20.1
NR	3589	3593	3594	3594

Table 5.11: Per frame and total running times for $Tracker1, 3$ and some of the OpenCV trackers for the sequence $ob090-03$. RT and PFR are the total running-time and the running time per frame, respectively.

	Tracker		Boost	CSRT	MIL
	1	3			
RT (min)	1.791	3.389	1.693	2.039	4.851
PFR(fps)	33.5	17.7	35.4	29.4	12.4

5.2.3 A Case Study

This subsection presents the results obtained for the *cows video*, although the *cows video* setup is different from the intended application of wildlife monitoring using drones. Indeed, the video configuration suggests that filming was done with a static camera from a side-view instead of at a high-altitude. Nevertheless, the *cows video* is used for testing to show how the proposed VOT solution works in practice, in particular, in handling occlusion. The *cows video* shows the occlusion of a cow (that is relatively static) by another cow, which occludes the former in several frames. It is interesting to track the occluded cow to get a sense of how the proposed framework works in such situations. Indeed, it is not excluded that the target stops for a moment to rest, eat, etc. It is important to consider this kind of situation primarily because the animal's motion is the *mirror* of the drone's motion.

Similar to the previous section, this section is divided into sub-parts which are introduced by headers. They indicate the subject in these sub-sections. These subjects are the introduction to new trackers, the reasons why the precision plot is not utilized for the *cows video*, the methodology used to manually handle performance evaluation on the *cows video*, the results obtained, the interpretation of the results produced on the one hand by the OpenCV trackers and on the other hand by the proposed VOT solution, the results for the resampling and finally the results for some of the trackers' speeds. Also, note that this subsection shows a number of frame sequences that are interpreted qualitatively.

Introduction to New Trackers

Similar to the videos in the MPI-AB dataset, tests were made for several trackers which are the *Tracker*1, 2*a*, 2*b*, 3, the OpenCV trackers, and new trackers addressed as *Tracker*7, 8, 9 and 10. They resemble *Tracker*2*a*,2*b*, 4, 5, 6 in that they are the PF with two measurement providers the colour image segmentation technique or either KCF, Median Flow, GOTURN or TLD, respectively, based on the SSIM index values (see Table 5.12). A number of particles $N = 2000$ are employed here as the targeted cow has a size much more important compared to the previous sequences.

Reasons for not Using the Precision Plot for the cows video

For this video, the precision plot is not a suitable performance for the following reasons:

- In this video, the animals are larger than the animals in the MPI-AB dataset. A tracker's estimate of the animal position can be located at any body-part of the

Table 5.12: Components for *Tracker*7, 8, 9, 10 which are all based on the PF. CIS is the abbreviation for colour image segmentation. The second measurement provider is called when the current SSIM value is below the set threshold.

Trackers	Measurement provider(s)	SSIM
*Tracker*7	CIS and KCF	Yes
*Tracker*8	CIS and Median Flow	Yes
*Tracker*9	CIS and GOTURN	Yes
*Tracker*10	CIS and TLD	Yes

latter. Given two trackers at a time step k, the distance between their respective outputs for the animal's location can be quite important (relative to a similar situation with the MPI-AB dataset) even if both trackers can locate the target animal. For this reason, it is also not suitable here to select as the ground truth, the outputs of the most successful tracker implemented in OpenCV. Even with the actual ground truth, the precision plot might still not be adapted although the location estimate errors should be less important than would be the case if the most accurate OpenCV tracker's outputs were selected as the ground truth.

- The occlusion happens along the x-axis, towards the end of the frames, and approximatively at half the total duration of the video sequence. Therefore, the precision plot cannot help to distinguish a tracker that shifted to the occluding animal from a tracker which momentarily lost the target during the occlusion, especially if it gave false positives along the y-axis. By similar reasoning, the ability of a tracker like TLD to handle this specific occlusion cannot be captured by the precision plot if such a tracker has missed the target detection in some frames before the occlusion.

To quantitatively evaluate the trackers' performance on handling the occlusion in the *cows video*, a manual count approach is employed. The manual count is performed on the second half of the *cows video* since the occlusion does not happen in the first frames and most trackers do not lose the target in these first frames. This sub video sequence contains 459 frames. The reader is invited to consult Appendix B for a detailed explanation about the methodology used to count the number of frames where a given tracker does not loose the occluded cow. A tracker is declared successful in handling the occlusion in the *cows video* if the number of frames where this tracker keeps tracking the target is above a threshold number. The latter is obtained by counting the number of frames the occlusion lasts in the sub video sequence.

Note that some of the trackers' video outputs do not have all of the 459 cropped frames. It is due to the OpenCV function used in Python to represent the trackers' outputs on the initial video. Indeed, this function was not always able to produce all 459 frames for the trackers' outputs. Yet, the maximal number of frames which were not written in these video streams is 7. It would not significantly affect the results given that for each tracker, it the ratio between their detections and their total frames that is considered.

Results for the Manual Count for the Trackers

Tables 5.13 and 5.14 report the results for the successful and unsuccessful trackers, respectively. In these tables, TD, TF, and DR are the total number of detections, the total number of frames, and the ratio of detections, respectively.

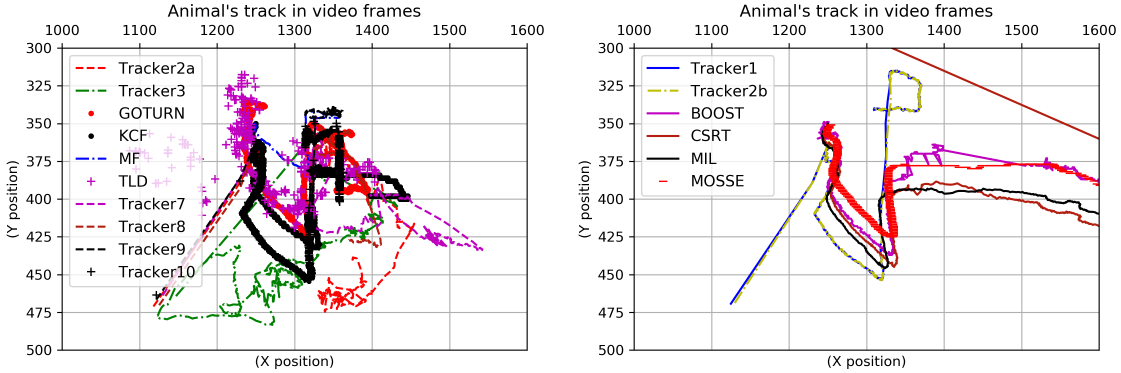
Table 5.13: Results for the successful trackers on handling the *cows video*'s occlusion case. TD, TF, and DR are the total number of detections, the total number of frames, and the ratio of detections, respectively.

	Tracker						KCF	MF	GO-TURN	TLD
	2a	3	7	8	9	10				
TD (frames)	150	372	199	403	426	410	355	449	418	184
TF (frames)	459	454	459	459	459	459	459	459	456	459
DR	0.327	0.819	0.434	0.878	0.928	0.893	0.773	0.978	0.917	0.401

These results are consistent with the trackers' trajectories shown in Figure 5.14. The left sub-figure presents the successful tracks, while the right one presents the failed tracks. It can be observed that most of the trackers that failed followed the passing cow, up to the moment where the latter started exiting the last frames. CSRT's trajectory went back to the frame's origin when the moving cow started exiting the video. It is not clear why it happened. *Tracker*1 and *Tracker*2*b* did not follow the moving cow, due to the colour image segmentation which provided faulty measurements that were located above the tracked cow. None of the successful trackers followed the occluding cow. Although their trajectories are not regular since they were also confused by the occlusion, it can still be seen that they are similar.

Table 5.14: Results for the unsuccessful trackers on handling the *cows video*'s occlusion case. TD, TF, and DR are the total number of detections, the total number of frames, and the ratio of detections, respectively.

	Tracker		Boost	CSRT	MIL	MOSSE
	1	2b				
TD (frames)	14	17	63	60	57	60
TF (frames)	452	459	459	459	459	459
DR	0.031	0.037	0.137	0.131	0.124	0.131



(a) Successful tracks.

(b) Failed tracks.

Figure 5.14: Trackers' trajectories for the static and occluded cow in the *cows video*. All trackers in the left sub-figure stayed on the target when the occlusion happened, while apart from *Tracker*1 and 2*b*, all other trackers followed the occluding cow. CSRT's tracking went back to the frame's origin when the occluding cow started exiting the last frames. *Tracker*1 and 2*b* started tracking a false positive located above the target. They could not recover, primarily due to the use of the anchor.

Interpretation of the Results for the OpenCV Trackers

Boosting and MIL failed in handling the occlusion since they learn the target's appearance online and use samples around the previously estimated location for training background/foreground classifiers. It seems that they tend to *forget* the initial target's appearance since they update their weak classifiers by using training examples extracted from the current estimated location. This suggests that Boosting and MIL adapt rapidly to changes in the target's appearance. CSRT also tracked the passing cow after the occlusion. It might be because of the involvement of the channel reliability maps. It is possible that the CF did not benefit from the colour channels of the regions which contain the two cows since CSRT normalizes the channels' contributions. Indeed, since both cows have similar colours, the channel which makes the difference between the cows might have been inhibited.

In contrast, TLD was able to accurately distinguish the target from the occluding cow. Some of the false positives observed with TLD were due to the detection of a cow which resembles the target and which is grazing at the beginning of the frames. KCF seems to learn good representations of objects when they are more or less static.

GOTURN and Median Flow handled the tracking well. This was unexpected given that, the former can fail in cases of long-term occlusions [12] (which is typically the case for the *cows video*) while the latter's keypoints tend to scatter during occlusions [95]. However, it is mentioned in [12] that when GOTURN handles an occlusion it is because the network *remembered* the target due to the use of the previous frame. The success of Median Flow might be explained by the lack of motion observed with the target and the resemblance in the two cows' shapes. Indeed, the majority of the keypoints used by Median Flow to detect the target should have been the same as the ones detected for the second cow during the full occlusion.

Interpretation of the Results for the Trackers which are based on the Proposed Tracking Framework

*Tracker*2*a*, 7, 8, 9, 10 succeeded so well because the SSIM index permits the occluding cow to be distinguished from the initially-tracked cow. Indeed, in this case, the range of values given by the SSIM index was between 0.48 and 0.75. The reason for such a good range of values is related to the lack of motion of the tracked cow and its size. The latter reduces the impact of the uncertainty around the anchor on the SSIM values compared to that of a small target. One might improve the PF tracking framework with relay between different measurement providers by proposing a similarity index less sensitive

to translation-related changes and changes in scale.

In the following, frame sequences generated from *Tracker*1, 2*a*, KCF, and *Tracker*7 are presented on single runs. The frames are taken at interval 13 frames, starting from the 518^{th} frame of the *cows video*. Similar to Figure 5.8, outputs from single runs are used because one needs to refer to the bounding boxes and to the ROI to interpret qualitative results. For these trackers, the SSIM values are also displayed on top of their frames. A value of 0.6 is also used as SSIM threshold for these tests.

For figure 5.15, *Tracker*1 was tracking the target while the passing cow was approaching. As long as the occlusion started, it lost the target for a false positive located above the initial target. Since *Tracker*1 relies on colour cues, it is possible that the proximity of the occluding cow produced a shadow which resulted in an illumination change. *Tracker*1 got trapped by this false positive above the target and could not recover due to the anchor.

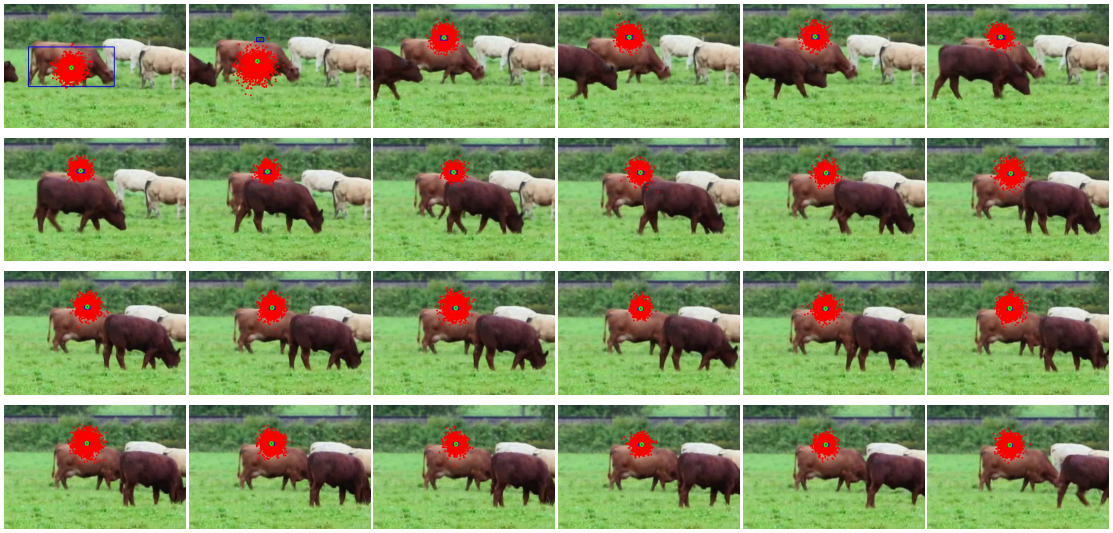


Figure 5.15: Example of *Tracker*1 failure in handling the occlusion of the *cows video*. Frames are displayed from left to right and from top to bottom starting with the 518^{th} frame with interval 13 frames. The SSIM index detected a change as soon as the occluding cow started passing in front of the target cow. *Tracker*1 started tracking a false positive above the target and could not recover because of the anchor involvement.

In figure 5.16, it can be seen that all of the frames with the SSIM index below the value 0.6 contain a red box. It is the ROI where YOLOv3 is supposed to detect the target. Some of these red boxes overlap with the bounding box produced by YOLOv3 while others do not. The latter case means that YOLOv3 was not able to detect the target. This case

could have resulted in the failure of *Tracker2a* primarily if the PF was not used. Indeed, it is due to the use of the particles to estimate the target's position that the tracking did not fail when there was a missed detection. In addition, in the proposed tracking framework, the last measurement is used at the current time step if no new measurement arrives. Yet, if the missed detections persist, the only chance for the tracker to recover is the presence of the SSIM index to call for the colour image segmentation. This suggests that the proposed framework is robust to occasional missed detections but can fail if no reliable measurements arrive.

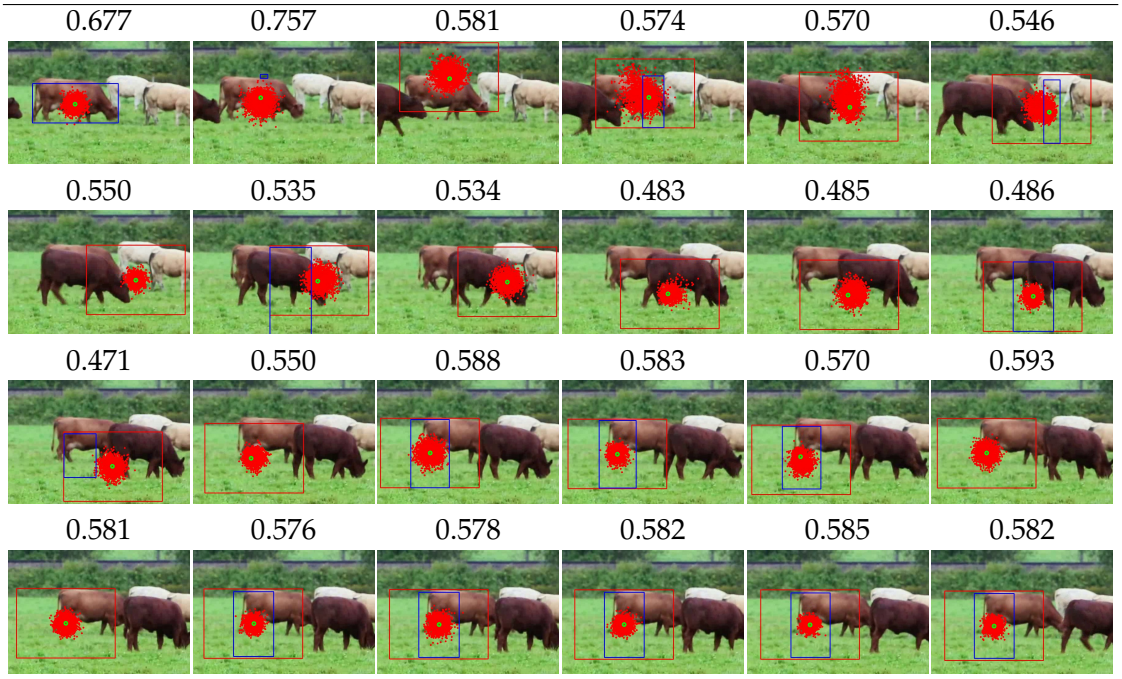


Figure 5.16: Example of occlusion handling with *Tracker2a* on the *cows video* for a SSIM threshold of 0.6. Frames are displayed from left to right and from top to bottom starting with the 518^{th} frame with interval 13 frames. Above each frame is the corresponding SSIM value. These values are less than the set threshold in most frames displayed. This justifies the call of YOLOv3. Yet, YOLOv3 is not able to detect the cow in some of these frames. The head and the feet of the cows are salient features for YOLOv3.

The frame at the 2^{nd} row and 6^{th} column shows that YOLOv3 detected the passing cow, while in the following frame it was able to get back to the initial target. It is not clear why YOLOv3 was able to detect the targeted cow. But it is possible that the targeted cow's features were more salient than the features of the passing cow, especially inside the ROI. Indeed, one outlier case where *Tracker2a* followed the passing cow was

observed. The estimate varies from one run to another due to the noise in the motion model. For the outlier case, the ROI might have been centered at an anchor which position favoured the detection of the passing cow by YOLOv3 rather than the targeted cow as observed for the currently presented example and for the averaged results presented earlier.

Nevertheless, this example shows on the one hand the importance of the anchor in the framework. On the other hand, it also shows that it is important to use for the second measurement provider a method able to accurately detect the initially tracked animal when the SSIM index signals a potential change if one is more interested in tracking a specific animal than a group of similar animals (i.e. animals which have similar colour and shape). YOLOv3 could be explicitly trained on the animal to monitor. Yet, YOLOv3 is trained to recognize a cow in general. It explains why YOLOv3 detects primarily the cow's head and feet which are salient features. It impacts YOLOv3's location estimate which tends not to be the animal's center. This justifies the importance of using a method different from the precision plot to evaluate the trackers' performance for the *cows video* and its occlusion.

The results presented in Figure 5.17, as well as the ones obtained for *Tracker*8, 9 and 10, support the importance of using for the second measurement provider in the proposed tracking framework, a method able to accurately recognise and localize the target. In the second last frame of this sequence, the SSIM value of 0.614 and the large bounding box (due to both cows segmented as a single object) indicate that the measurement is furnished by the colour image segmentation. In subsequent frames, KCF was called and able to successfully find the target by dint of the latter's discriminative features and/or lack of motion. Figure 5.18 shows successful tracking by KCF.

Note that the occluding cow has been tracked also with some of the previous trackers. For clarity, results and analyses are presented in Appendix B.

Tables with Statistics (about SSIM, Running Times and Others) for some of the Trackers for the occluded cow

Tables 5.15 and 5.16 present other results for some trackers for the *cows video*. For the trackers with as the second measurement provider, one of the OpenCV trackers, the OpenCV tracker's name is reminded in brackets.

Table 5.15: Data for the trackers based on the proposed tracking framework for the *cows video*. The *cows video* has 980 frames in total. The SSIM threshold is 0.6. SSIM index statistics are between 0 and 1, given that the absolute value of the SSIM index is considered here. NS, NDS, RT, PFR, and NR are the number of frames where the SSIM values were below the set threshold, the number of actual detections in these frames, the total running time, the running time per frame, and the number of times the particles have been resampled.

	Tracker					
	2a	2b	7 (KCF)	8 (MF)	9 (Goturn)	10 (TLD)
NS (frames)	419	83	253	413	312	251
NDS (frames)	234	11	253	413	312	251
SSIM Mean	0.622	0.639	0.637	0.629	0.644	0.638
SSIM STD	0.061	0.038	0.041	0.049	0.052	0.042
SSIM Max	0.742	0.740	0.745	0.748	0.752	0.749
SSIM Min	0.487	0.566	0.571	0.544	0.547	0.550
RT (min)	3.159	1.184	1.044	0.79	1.688	2.99
PFR (fps)	5.2	13.8	15.7	20.7	9.7	5.5
NR	440	454	410	456	457	403

Table 5.16: Per frame and total running times for *Tracker*1, 3 and the OpenCV trackers used in *Tracker*7, 8, 9, 10, respectively for the *cows video*. RT and PFR are the total running-time and the running time per frame, respectively.

	Tracker		KCF	MF	GOTURN	TLD
	1	3				
RT (min)	0.433	6.37	0.440	0.264	1.168	2.412
PFR(fps)	37.7	2.6	37.1	62.8	13.9	6.8

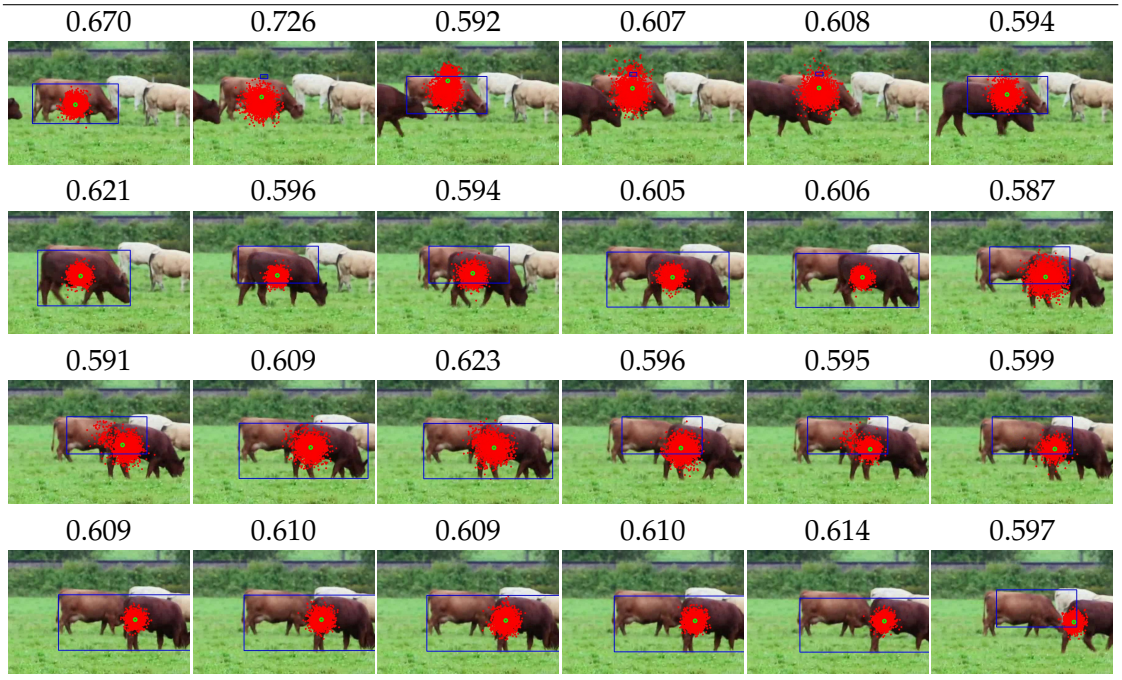


Figure 5.17: Example of occlusion handling with *Tracker*7 on the *cows video* for a SSIM threshold of 0.6. Frames are displayed from left to right and from top to bottom starting with the 518^{th} frame with interval 13 frames, apart from the last two frames which have interval 14 frames. Above each frame is the corresponding SSIM value. In some of the frames, the measurements from the colour image segmentation are above the target. But, *Tracker*7 did not get stuck tracking the false positive above the target because KCF provided reliable measurements.

Results for Particle Resampling

There is resampling almost every two frames. That is half what was observed for the sequences of the MPI-AB dataset for the threshold $2N/3$. It is because the same value is utilized as the resampling threshold. It corresponds here (for the *cows video*) to the value $2N/30$. Tests were also repeated for the occluded cow with the trackers that involve the PF with a threshold of $2N/600$. A lower frequency of resampling is observed. It is 5 to 4 times less than the frequency observed with the threshold $2N/30$ with similar performance. This confirms that the resampling threshold was the source of such a frequent resampling for $2N/30$. It also suggests that the performance of the proposed VOT solution might not be affected by the number of resampling, if the set threshold enables a minimum number of resampling. It is probably due to the use of the anchor and the two measurement providers.

Figure 5.18: Example of occlusion handling by KCF on the *cows video*. Frames are displayed from left to right and from top to bottom starting with the 518^{th} frame with interval 13 frames. KCF successfully handled the occlusion.

Fastest Trackers

When tracking the relatively static cow, *Tracker*9 and 10 are more suitable than the other trackers with the same framework for the *cows video*. But *Tracker*8 is faster than *Tracker*9 and *Tracker*10 due to Median Flow. Median Flow uses a basic way to represent and track the target relative to the other trackers that involve costlier techniques in terms of computation. Similar to the observation made for the MPI-AB sequences, *Tracker*8 benefited from the running speed of Median Flow.

5.3 Handover

This section firstly simulates the communication protocol designed for the handover. Secondly, this section presents the mechanism by which *DroneB* verifies if it locked its camera on the target tracked by *DroneA*.

5.3.1 Communication Protocol Simulations

The following tables present simulations of the proposed handover communication protocol (presented in Subsection 4.3.3 of the previous chapter) with $\beta = 1, n_A = 3, n_B = 3$ and therefore $\Delta_B = k', \Delta_A = k'$. Recall that k' is the time step for the handover communication protocol. It is defined by $k' = k + \alpha\zeta$ with $\alpha = 0, 1, 2, 3, \cdots$ and $\zeta = 1/\zeta'$, $\zeta' \in \mathbb{N}$.

For clarity, the messages $K_A\rho$, $K_BK_A\rho$ and $K_AK_BK_A\rho$ are represented by ①, ② and ③, respectively. The syntax "Wait (3 | 1)" means that *DroneB* is waiting for either ① or ③ since there is no guarantee that its previous message was delivered to *DroneA*. The handover communication protocol begins after both drones have aligned at a given time step k of the target's tracking by *DroneA*. The variable k' can then be initialised with this time step k.

Table 5.17 shows results in the case of a perfect exchange, i.e. without any message loss or any drone inability to receive or send a message. The handover's outcome is positive since *DroneA* was able to hand the tracking to *DroneB* and to return home. The whole process took only four-time steps, with regards to the chosen $\Delta_A = \Delta_B = k'$.

All remaining tables present cases where messages got lost at some point in the handover communication protocols. Note that although the study refers only to the absence of message as message loss, these tables are also valid in the cases of non-emission and non-receipt of messages, because for the drone-receiver all three situations appear as a message loss situation.

Table 5.17: No loss of message. The handover is successful.

Time steps	*DroneA*'s actions	ctr_D	n_{TA}	w_A	**Message states**	w_B	n_{TB}	lev	*DroneB*'s actions
$k' = k$			0	0		0	0	1	
$k' + \zeta$	Send ①	1	1	0	① : Sent	1	0	1	Wait ①
$k' + 2\zeta$	Wait ②	1	1	1	① : Delivered ② : Sent	0	0 1	2 2	Receive ① Send ②
$k' + 3\zeta$	Receive ② Send ③ *Return*	Kill	0 1	0	② : Delivered ③ : Sent	1	1	2	Wait ①
$k' + 4\zeta$	Pass				③ : Delivered	0	0	3	Receive ③ *Track*
$k' + 5\zeta$									Pass

In Table 5.18 the first message sent at time $k' + \zeta$ by *DroneA* was lost. Since no drone knows about this fact, *DroneA* waited to receive *DroneB*'s acknowledgement while *DroneB* waited to receive the first message. Conditions about the closure of *DroneA*'s receptive window and the maximal number for sending a *level* 1 message permitted *DroneA* to retry to send its message. *DroneB* waited for three-time steps as a result of

the constraint about the maximal time of the receptive window for *DroneA*'s first message, i.e. $n_A(\beta+1) = 6$ here. Since *DroneA*'s second attempt was successful and no more messages were lost, *DroneA* was able to return home at the end of the handover while *DroneB* continued the tracking. Hence, the first message loss did not compromise the positive outcome of the handover although it increases the total duration by two more steps relative to the perfect exchange presented in Table 5.17. One of them was used by *DroneA* to wait for *DroneB*'s response while the second was for *DroneA*'s first attempt. This shows the impact of both the duration of the drones' receptive windows and the maximum number of message-sending tries on the handover total duration.

Table 5.18: The first message got lost the first time it was sent. Handover is successful because *DroneA* can try to send again its first message after waiting a certain time. The second try was delivered successfully.

Time steps	*DroneA*'s actions	ctr_D	n_{TA}	w_A	**Message states**	w_B	n_{TB}	lev	*DroneB*'s actions
$k'=k$			0	0		0	0	1	
$k'+\zeta$	Send (1)	1	1	0	(1): Lost	1	0	1	Wait (1)
$k'+2\zeta$	Wait (2)	1	1	1		2	0	1	Wait (1)
$k'+3\zeta$	Send (1)	1	2	0	(1) : Sent	3	0	1	Wait (1)
$k'+4\zeta$	Wait (2)	1	2	1	(1) : Delivered (2) : Sent	0 0	0 1	2 2	Receive (1) Send (2)
$k'+5\zeta$	Receive (2) Send (3) *Return*	Kill	0 1	0	(2): Delivered (3) : Sent	1	1	2	Wait (3 | 1)
$k'+6\zeta$	Pass				(3) : Delivered	0	0	3	Receive (3) *Track*
$k'+7\zeta$									Pass

In the scenario of the impossibility of having the first step of the handover communication protocol completed as presented in Table 5.19, the total duration is eight-time steps. The first six-time steps correspond to the total waiting time of *DroneB*. This suggests that the number of *DroneA*'s tries should be carefully tuned. However, it also gives more chances to *DroneA* to be relayed. Although the protocol's outcome was not positive in this case for the first *DroneB* candidate, the availability of the possibility for calling another *DroneB* could result in the handover success.

Table 5.19: The first message got lost three times successively. Handover did not lead to success yet, i.e. with the first *DroneB* candidate. *DroneB* waited for a total of $n_A(\beta+1) = 6$ and entered the mode *Return* while *DroneA* was calling another *DroneB* candidate.

Time steps	*DroneA*'s actions	ctr_D	n_{TA}	w_A	**Message states**	w_B	n_{TB}	lev	*DroneB*'s actions
$k' = k$			0	0		0	0	1	
$k' + \zeta$	Send (1)	1	1	0	(1) : Lost	1	0	1	Wait (1)
$k' + 2\zeta$	Wait (2)	1	1	1		2	0	1	Wait (1)
$k' + 3\zeta$	Send (1)	1	2	0	(1) : Lost	3	0	1	Wait (1)
$k' + 4\zeta$	Wait (2)	1	2	1		4	0	1	Wait (1)
$k' + 5\zeta$	Send (1)	1	3	0	(1) : Lost	5	0	1	Wait (1)
$k' + 6\zeta$	Wait (2)	1	3	1		6	0	1	Wait (1)
$k' + 7\zeta$	Call New *DroneB*	2	0	0					*Return*
$k' + 8\zeta$	Pass								Pass

The case shown in Table 5.20 demonstrates that the loss of *DroneB*'s response hardly results in the handover failure due to the fact that *DroneB* can also send messages n_B times at most. The last case considered, i.e. Table 5.21, confirms that it is not enough that both drones exchange *level* 1 and *level* 2 messages, because it is important that the third message is delivered to *DroneB*. Indeed, it is necessary that *DroneB* has the confirmation that *DroneA* is returning home and is not calling another *DroneB* candidate.

5.3.2 Simulation of the Method for the Target Identity Validation

The mechanism for the validation of the identity of the target during handover (steps $11-15$ of Algorithm 3) is simulated for all the video sequences, under the assumption that the drones involved have the same altitude and the same orientation. These assumptions are realistic because one can employ the drones' GPS (or another global localization system) to correct the drone-peer's position relative to the drone-tracker's one. As a reminder, the location where the handover happens in the field should be selected carefully by the drone-tracker to promote the strength of the GPS signal. This choice could be achieved by the drone-tracker based on information received from sensors able to detect surrounding obstacles (trees for example). The two drones' orientations for handover could be preset. It is important that these conditions are met for the relay of the tracking because the drone-tracker and the drone-peer only rely on the

Table 5.20: The second and third messages got lost. The handover was successful because *DroneB* could re-send its message, which was well-delivered to *DroneA*.

Time steps	*DroneA*'s actions	ctr_D	n_{TA}	w_A	**Message states**	w_B	n_{TB}	lev	*DroneB*'s actions
$k' = k$			0	0		0	0	1	
$k' + \zeta$	Send ①	1	1	0	① : Sent	1	0	1	Wait ①
$k' + 2\zeta$	Wait ②	1	1	1	① : Delivered ② : Lost	0	0 1	2 2	Receive ① Send ②
$k' + 3\zeta$	Send ①	1	2	0	① : Lost	1	1	2	Wait 3 | 1
$k' + 4\zeta$	Wait ②	1	2	1	② : Sent	0	2	2	Send ②
$k' + 5\zeta$	Receive ② Send ③ *Return*	kill	0 1	0	② : Delivered ③ : Sent	1	2	2	Wait 3 | 1
$k' + 6\zeta$	Pass				③ : Delivered	0	0	3	Receive ③ *Track*
$k' + 7\zeta$									Pass

Table 5.21: *DroneA*'s last message got lost. The handover was not successful because both drones stopped tracking the target. The delivery of the last message is crucial for the tracking continuation.

Time steps	*DroneA*'s actions	ctr_D	n_{TA}	w_A	**Message states**	w_B	n_{TB}	lev	*DroneB*'s actions
$k' = k$			0	0		0	0	1	
$k' + \zeta$	Send ①	1	1	0	① : Sent	1	0	1	Wait ①
$k' + 2\zeta$	Wait ②	1	1	1	① : Delivered ② : Sent	0	0 1	2 2	Receive ① Send ②
$k' + 3\zeta$	Receive ② Send ③ *Return*	Kill	0 1	0	② : Delivered ③ : Lost	1	1	2	Wait 3 | 1
$k' + 4\zeta$	Pass					0	2	2	Send ②
$k' + 5\zeta$						1	2	2	Wait 3 | 1
$k' + 6\zeta$						0	3	2	Send ②
$k' + 7\zeta$						1	3	2	Wait 3 | 1
$k' + 8\zeta$									*Return*
$k' + 9\zeta$									Pass

information that they capture with their cameras since no tracking devices are attached to the target.

The study considered that FOV_A during a handover is the 200^{th} frame for sequences $ob002-01a$, $ob002-01b$ and the *cows video*, the 1000^{th} frame for the sequence $ob003-01$, the 600^{th} frame for the sequence $ob090-03$. Before selecting these specific frames, it was ensured that the target's location estimate was reliable as the outputs produced by $Tracker2a$ are used. In Algorithm 3, there is a step which permits *DroneA* to send the relevant information to *DroneB*, only if there is a minimal similarity between the initial template and the current target's appearance.

In order to derive FOV_B, all of the pixels in x and y directions have to be translated by tx and ty, respectively. In the ideal scenario depicted in Figure 4.5, tx is proportional to the real distance separating the two drones along the horizontal axis while ty is zero as the drones alignment is perfect along the vertical axis. However, for the simulations to better reflect real-life scenarios, all of the pixels are instead translated by $tx' = tx + \gamma$ and $ty' = ty + \gamma$ where γ is zero-mean Gaussian white noise since the drones' *FOVs* are not always stable during a flight. Figure 5.19 shows a simulation of both drones' *FOVs* for the sequence $ob002-01b$.

(a) *DroneA*'s *FOV*. (b) *DroneB*'s *FOV*.

Figure 5.19: *FOVs* simulated for both drones for the sequence $ob002-01b$ at $k = 200$. The original images are cropped for more clarity. *DroneA* has a better perspective of the left side of the scene because it is at the left of *DroneB* (see Figure 4.5). Similarly, *DroneB* has more information on the right side of the scene. A simple way to understand both drone's *FOVs* operation is to make an analogy with how human-eyes work.

Furthermore, note that tx and γ can be empirically determined for real-world applications. Three methods have been tested to compute the SSIM index to be compared to a threshold value in order to determine the success of the handover. Indeed, the objective is to identify the most suitable approach amongst the three that provide the best

location for the target in the field of view of the drone-peer. This location corresponds to the initial anchor of *DroneB*'s tracking (if the handover is successful).

- For the first method, each particle $x_k^i(a_k^i, b_k^i)$ is translated using the computed translation factors tx and ty, with in addition a noisy component also drawn from γ, to obtain the translated particles $x_k^{i\prime}(a_k^{i\prime}, b_k^{i\prime})$, as shown in the following equation:

$$\begin{cases} a_k^{i\prime} \leftarrow a_k^i + t_x + \gamma_{k,x}^i \\ b_k^{i\prime} \leftarrow b_k^i + t_y + \gamma_{k,y}^i \quad with\ x_k^i(a_k^i, b_k^i). \end{cases} \tag{5.3.1}$$

  In this equation, the variables $\gamma_{k,x}^i$ and $\gamma_{k,y}^i$ drawn from γ are added to the particles' components. Once again, it is because the drones' *FOVs* are not perfectly stable and the exact value of the noise, that add to the computed distance (from tx, ty) between the elements in the *FOVs* of the two drones is not known. Next, *DroneB* computes an estimate of the animal's centre location using *DroneA*'s translated particles and their weights:

$$\hat{x}_k' = \sum_{i=1}^{N} w_k^i x_k^{i\prime} \quad with\ x_k^{i\prime}(a_k^{i\prime}, b_k^{i\prime}). \tag{5.3.2}$$

  An ROI surrounding this estimated location can then be extracted from FOV_B and compared to the ROI (with same dimensions) around the animal in FOV_A with the SSIM index.

- For the second method, similar to the first method, the positions of the translated particles are computed (Equation 5.3.1). Next, the SSIM measure is determined between the ROI in FOV_A and the ROI with centre location a translated particle $x_k^{i\prime}(a_k^{i\prime}, b_k^{i\prime})$. The centre of the ROI which produces the highest SSIM value can then be used to update the estimate of the tracked animal's position in FOV_B.

- The third method follows the same steps as the previous, except that the particles used here are new ones, $x_k^{i\prime\prime}(a_k^{i\prime\prime}, b_k^{i\prime\prime})$. They are generated by adding Gaussian white-noise γ' to the estimate of the animal's location as shown in the following equation:

$$\begin{cases} a_k^{i\prime\prime} \leftarrow \hat{a}_k' + \gamma_{k,x}^{i\prime} \\ b_k^{i\prime\prime} \leftarrow \hat{b}_k' + \gamma_{k,y}^{i\prime} \quad with\ \hat{x}_k'(\hat{a}_k', \hat{b}_k'). \end{cases} \tag{5.3.3}$$

Note that the indices x and y in $\gamma_{k,x}^{i\prime}$ and $\gamma_{k,y}^{i\prime}$ indicate the directions in which the latter are applied. The animal's location is also estimated similarly to what is performed for the first method (see Equation 5.3.2).

For the MPI-AB sequences, results for 100 tests per method, $tx = 200$ pixels, $ty = 10$ pixels, $\gamma \sim \mathcal{N}(0, 6^2)$, $\gamma' \sim \mathcal{N}(0, 8^2)$ and a $m \times m$ window for the ROI are reported in the following tables, where the time corresponds to the running time of a single test. Table 5.22 shows results for the MPI-AB sequences for the three methods with the same number of particles which is the initial number (200) used by *DroneA* to track the animal up to the handover. Table 5.23 only shows results for different values of the number of particles (NP) for the third method. Tables 5.24 and 5.25 contain results for the *cows video* in the same conditions apart from the number of tests which is 20 as a more important number of particles (2000) is used here.

Note that Figures 5.20 and 5.21 also show some graphical examples for the three methods for sequence $ob002 - 01b$. The examples selected are representative of the average behaviour for a given method.

The first method performs worse compared to the others (see Tables 5.22 and 5.24). It is because of the uncertainty which adds to the computed distance between elements in the two drones' $FOVs$. Apart from the *cows video*, for this method, the SSIM index lies in a small range of values similar to the ranges observed for the tests of the VOT solution on the sequences $ob002 - 01a$, $ob002 - 01b$, $ob003 - 01$ and $ob090 - 03$. It confirms that the SSIM index is sensitive to pixel precision, i.e. to the translation in the positions of animals to compare, especially when they have a small scale.

The second and third methods perform similarly at an equal number of particles since they both make use of multiple hypotheses drawn around the PF estimate. The same reasoning explains why with more particles when using the third method, the SSIM index also increases (Tables 5.23 and 5.25). However, the gain in performance negatively affects the running time, especially in the case of the *cows video*. One good compromise would be to use the third method with a relatively high number of particles for small targets and to use the second method for medium/large scale targets. Yet, the use of the first method is not excluded for targets that have a large scale.

5.4 Discussion

The results for the proposed tracking framework on the MPI-AB dataset show that YOLOv3 is not convenient as the second measurement provider when the animals appear small from high altitudes because YOLOv3 misses detections. It was the case for most of the frames where YOLOv3 was called. This problem is worsened by the use of the ROI. Yet, the ROI is important here because it is meant to further reduce YOLO's running time and to mitigate its inability to accurately distinguish the background from

Table 5.22: SSIM index statistics for 100 tests and 200 particles for the three methods (MTD). The range of values for the SSIM index is between -1 and 1.

	ob002-01a			*ob002-01b*			*ob003-01*			*ob090-03*		
MTD	1	2	3	1	2	3	1	2	3	1	2	3
Mean	0.21	0.58	0.64	0.14	0.62	0.70	0.04	0.57	0.59	0.35	0.77	0.73
STD	0.01	0.21	0.22	0.02	0.20	0.17	0.01	0.23	0.23	0.04	0.14	0.13
Max	0.24	0.97	0.97	0.17	0.83	0.83	0.08	0.97	0.97	0.42	0.99	0.99
Min	0.19	0.26	0.28	0.10	0.23	0.24	0.02	0.25	0.24	0.28	0.45	0.44
Time(s)	.007	0.9	0.93	.005	0.6	0.6	.005	0.6	0.6	.005	1.	1.

Table 5.23: SSIM index statistics for 100 tests for Method 3 with a different number of particles. The range of values for the SSIM index is between -1 and 1.

	ob002-01a			*ob002-01b*			*ob003-01*			*ob090-03*		
NP	400	600	800	400	600	800	400	600	800	400	600	800
Mean	0.75	0.80	0.85	0.78	0.82	0.82	0.70	0.75	0.81	0.86	0.89	0.89
STD	0.20	0.17	0.14	0.10	0.02	0.05	0.21	0.18	0.17	0.12	0.10	0.10
Max	0.97	0.97	0.97	0.83	0.83	0.83	0.97	0.97	0.97	0.99	0.99	0.99
Min	0.28	0.31	0.56	0.46	0.75	0.48	0.32	0.28	0.40	0.56	0.72	0.72
Time(s)	1.8	2.7	3.7	1.2	1.9	2.4	1.2	1.9	2.5	1.3	1.9	2.5

Table 5.24: SSIM statistics for 20 tests and 2000 particles for the three methods (MTD). The range of values for the SSIM index is between -1 and 1.

	cows video		
MTD	1	2	3
Mean	0.75	0.986	0.993
STD	0.03	0.01	0.004
Max	0.80	0.994	0.994
Min	0.71	0.953	0.98
Time (s)	0.03	50	49.9

Table 5.25: SSIM statistics for 20 tests for Method 3 with a different number of particles. The range of values for the SSIM index is between -1 and 1.

	cows video		
NP	4000	6000	8000
Mean	0.994	0.994	0.994
STD	0.	0.	0.
Max	0.994	0.994	0.994
Min	0.994	0.994	0.994
Time (s)	99.6	149.3	201.2

(a) *DroneA*'s *FOV*.

(b) *DroneB*'s *FOV* with the 1^{st} method. $SSIM = 0.148$.

(c) *DroneB*'s *FOV* with the 2^{nd} method. $SSIM \in [0.017, 0.590]$.

(d) *DroneB*'s *FOV* with the 3^{rd} method. $SSIM \in [0.040, 0.668]$.

Figure 5.20: *DroneA*'s *FOV* and *DroneB*'s *FOV* for the sequence $ob002 - 01b$ at $k = 200$ for the three methods with $N = 200$ particles which are represented in red. For a subfigure, the white ROI contains a particle (also in white) that is the less likely animal's location. The green particle (which has an ROI also in green) corresponds to the most likely location.

(a) *DroneA*'s *FOV*.

(b) *DroneB*'s *FOV* with $N = 400$ particles. $SSIM \in [0.040, 0.847]$.

(c) *DroneB*'s *FOV* with $N = 600$ particles. $SSIM \in [0.017, 0.902]$.

(d) *DroneB*'s *FOV* with $N = 800$ particles. $SSIM \in [0.044, 0.902]$.

Figure 5.21: *DroneA*'s *FOV* and *DroneB*'s *FOV* for the sequence $ob002 - 01b$ at $k = 200$ for the third method. $N = 400, 600, 800$ particles which are represented in red. For a sub-figure, the white ROI contains a particle (also in white) that is the less likely animal's location. The green particle (which has an ROI also in green) corresponds to the most likely location.

the foreground. The ROI acts like *blinkers* and helps YOLOv3 to focus on a small patch containing the target. In addition, YOLOv3 is unable to recognize the target when other animals come into the vicinity of the latter despite the use of the ROI. It is because YOLOv3 was not explicitly trained to recognize the target.

But, it might not help much to train YOLOv3 on the specific targets given their scale, unless the tracking framework is used to track a group of animals with similar features instead of a specific animal. It is generally the case for group of animals from the same species. For a human observer, it is also difficult to differentiate the animals from a high altitude, especially when they resemble to each others. One element which helps the human observer is that the latter has prior information of the target's trajectory. It is also one of the reasons why Boosting and CSRT were able to learn the target's appearance from such an altitude. Amongst the other reasons are the involvement by these methods

of specific features such as the orientation histograms, the Haar-like features and the channel and spatial reliability scores to model the target.

The results that support the importance of the target's previous trajectory for Boosting, CSRT and hence for *Tracker*4, 5, are those of the first sub-figure of Figures 5.10, 5.11, 5.12 and 5.13. In these cases, the performance of *Tracker*4 and *Tracker*5 were affected negatively because, Boosting and CSRT could not always provide reliable measurements to the PF in *Tracker*4 and *Tracker*5 since Boosting and CSRT were deliberately prevented from learning the target's appearance when the colour image segmentation was called by the SSIM index. Recall that this was performed to assess the robustness of Boosting and CSRT to missing data (missing frames) and hence the robustness of *Tracker*4, 5 to missing/uncertain measurements from the second measurement provider. CSRT missed detections in the video sequences $ob002 - 01a$ and $ob090 - 03$. The drop in performance for Boosting and CSRT in case of missing data, also shows why the ROI is not suitable in the proposed tracking framework when the second measurement provider is a kernel or a correlation-filter based tracker. Indeed, the use of the ROI induces imprecision in the locations where Boosting and CSRT expect to find the target, because the ROI is centered at the anchor which is uncertain.

For the above mentioned sub-figures (i.e. the first sub-figure of Figures 5.10, 5.11, 5.12 and 5.13), *Tracker*5 shows average performance, primarily because of the presence of the SSIM index and the colour segmentation technique. But, it is also due to the use of the previous measurement available and the PF when no new measurements are provided. In fact, without counting CSRT's number of detections in *Tracker*5's results (when CSRT cannot learn independently of the colour segmentation), one would not have known that most of the measurements were provided by the colour image segmentation.

Apart from in video $ob090 - 03$, *Tracker*1 performs quite well. However, it is better to consider using the proposed tracking framework because of the difficulty of *Tracker*1 when the foreground and background are similar and when there are many false positives in frames following the use of the colour segmentation. Ensemble techniques are more beneficial for tracking since different types of challenges can be observed for a video sequence and handled by a tracker more adapted to that challenge than the others in the ensemble. As a matter of fact, a given method can only outperform others on a given class of challenges. No Free Lunch Theorems (NFLTs) state that given a set of problems, an algorithm can only perform well on a subset while its performance deteriorates on the remaining problems [131; 132]. The second sub-figure of Figure 5.13 illustrates this fact for *Tracker*4 and 5 relative to *Tracker*1. Boosting also performed quite

well for the sequences in the MPI-AB dataset. But Boosting could not handle the occlusion case in the *cows video* (when the target was the cow with limited movements).

One can recommend *Tracker*4, 5 (with no missing data for Boosting and CSRT) for targets viewed from high altitude. *Tracker*9 could be more suitable with drones' footage recorded at a relatively low altitude and for targets that move slowly. Also (with *Tracker*9), GOTURN might require to be trained. For small targets with fast motions, KCF and Median Flow might not be adapted.

The results on the sequences from the MPI-AB dataset show that MIL's tracking is less precise than the ones for Boosting, CSRT, and some of the proposed trackers. But MIL performed better than the other OpenCV trackers. MIL tends to track the target's shadow. YOLOv3 also detects sometimes the target with its shadow. This suggests that the time of the day can impact the outcome of tracking. Furthermore, the daylight might influence the measurements of the colour image segmentation. To mitigate such influence, the frames captured by the drones could be preprocessed to remove the animals' shadows and adjust the illumination in the frames. One has to be aware of such challenges in the field.

*Tracker*2's framework and all of the other trackers which are based on the proposed tracking framework can handle intermittent missing and/or wrong data even if the latter can lead to location estimates which are less precise (i.e. which are close to the target's boundaries). Yet, they are sensitive to persistent wrong measurements (even with in between reliable measurements). When they handle missing measurements, it is possible due to the use of the previous measurement if no new observation is available and to the PF's convergence. Their inability to recover when they lose the target is aggravated by the presence of the anchor. Indeed, for *Tracker*2, all measurements are sampled around the anchor while for the others, the CIS keeps producing observations around the anchor.

With the MPI-AB video sequences, it was observed that the SSIM index is not adapted enough when used in the proposed VOT solution. It is not suitable for the tracking of animals in video frames captured by drones at high altitudes. The ranges of SSIM values observed with the MPI-AB videos compared to the *cows video* supports this observation even though the targeted cow also barely moves. It is also evidenced by the sensitivity of the SSIM index to pixel errors for small image patches observed with the validation of the target's identity during a handover when applying the first method. Moreover, it was observed that the second and the third method were more relevant for the confirmation of the target's identity during a handover because both methods involve multiple hypotheses where one can find the target in FOV_B.

One can improve the proposed tracking framework, in particular for small objects, by replacing the SSIM index with a method more sensitive to the target's changes in appearance. In this case, a detection module able to accurately detect the target is still required to correct the colour image segmentation. Indeed, when the similarity measure signals a change in appearance, the detector can bring the colour image segmentation back on the target after successive detections. This was the initial idea. Recall that successive detections will be needed because of the PF's convergence. Indeed, the PF needs more than one accurate measurement for the majority of the particles to displace to a certain location especially if this location is relatively far from the PF's actual position. It is the principle by which the PF can handle occasional imprecise or missing measurements. As mentioned before, the requirement of accurate initial measurements can be helped by restricting the area where the particles are initially generated. One might not need to use the PF if both the similarity index and the second measurement provider have very good performance.

For targets that frequently move or undergo different appearance changes, one can update the initial template periodically. But, instead of changing the template to a patch extracted in initial frames, a pool of templates containing multiple appearances of the target can be employed. For example, one can generate several templates by periodically rotating the initial template to capture changes in the pose and moreover scale some of these templates to represent changes in the scale. The former way of updating is not recommended because there is the risk that the new template would be taken when the target's appearance has completely changed (for example due to occlusion). Regarding the approach of a set of templates, at the template update time, the current appearance of the target could be compared to all of the templates in the pool and the one with the highest similarity could be selected as the new template. The range of SSIM index could benefit from having such a set of templates.

Furthermore, one can associate a subset of templates to a given OpenCV tracker (or any other tracker) in such a way that the most suitable tracker would be selected as the measurement provider, allowing more than two providers for the PF. For example, if the selected template represents a reduction in scale, a tracker like Boosting can be called while if the change is the augmentation in scale, a method like KCF can rather be used. However, since these methods tend to not accurately represent the target when there is missing data, one might limit their number to three or four for applications intended to run in real-time. A different ensemble approach could be employed rather than the tracking framework proposed with the trackers that performed the best in this study for tracking offline. Such an ensemble approach could exploit at a time step, the median of

the trackers' outputs.

The anchor is important in the proposed framework because the similarity measure depends on it as well as the location where the colour image segmentation can pick the next measurement and even the second provider if the latter is a detection module. It is at the location of the anchor that the relevant image patch is selected for comparison to the target's template. If the similarity measure is reliable the latter can help to correct the anchor. If not, the only chance for the anchor to not lead to the tracking failure is the presence of a second reliable measurement provider.

It was observed from the results that the scale of the target can be a challenge for the proposed VOT solution but especially for the SSIM index. A way to mitigate this when deploying the system in the real world would be to employ cameras with zooming features. It will help to increase the target's size on drones' footage without decreasing the latter's altitude. Yet, there is a risk of losing information in the images.

Regarding the handover communication protocol, the results of the simulations have shown that a handover hardly fails. This is primarily due to the possibility that the communication protocol offers in terms of maximum attempt numbers for sending a *level* 1 or a *level* 2 message, calling a new *DroneB* candidate, and stopping the exchange after a *level* 3 message is sent by *DroneA* as seen with Table 4.1. With Table 4.1, one can see that if a handover fails, with this communication protocol, the worst that could happen is that the tracking stops. Hence, it is not possible to have more than two drones involved in a handover if the handover fails.

The communication protocol also involves the time step k' and the variable β. In practice, the former can be selected with regards to the time required to send a message from *DroneA* to *DroneB* while β can be adjusted to the time required to check for the target to be relayed between the two drones using one of the three methods presented earlier. These methods have their pros and cons. One would have to consider the scale of the target to select the appropriate method. Indeed, the first method was only adequate for an animal with a large scale, while the second and the third method better fit small to medium-scale targets.

In practice, it is possible that the duration of the calculations involved in the step for the verification of the target's identity influence the tracking once the new drone-tracker takes over, in particular with the second and the third method. Yet, in order to prevent this, several recommendations can be used. For the third method, a higher value could be used for the factor γ' for spreading the particles to reduce the superposition of the particles. On the other hand, *DroneA* could assist *DroneB* in speeding the calculations required by the second and the third method. For example, the new particles' positions

in *DroneB*'s *FOV* could be computed by *DroneA* before message-sending to *DroneB* or the computations could be parallelized on both drones. The calculation of the SSIM index for the image patches could be performed in ascending order. Indeed, the first particles that would be considered would be the one close to the re-calculated PF estimate. In this way, the computations could be stopped just after finding the first particle with a SSIM index above the threshold value for the SSIM index. Additionally, as long as the handover is successful, the anchor can help *DroneB* to keep tracking the target while a trained detection module or a learning method could be used to readjust the PF location estimate in *DroneB*'s *FOV*. In the latter case, the new drone-tracker could benefit from the target's appearance learnt by the former drone-trackers.

In summary, the proposed tracking framework's performance is unsatisfactory when YOLOv3 is used for the second measurement provider, especially with small-scale targets. It is because YOLOv3 misses a significant number of detections for several reasons. The target and therefore the adaptive ROI have small sizes which do not facilitate the detection task for YOLOv3. The latter is also not explicitly trained on the animal to track. The results with *Tracker*2*b*, in particular, show that the proposed framework can handle occasional missing data and faulty measurements due to the use of the previous measurement and to the ability of the PF to converge relatively slowly to the measurements. However, in the case of persistent missing or faulty measurements (even if reliable measurements arrive in between), the PF might not recover because of the presence of the anchor. If it is expected that the first measurement provider, here the colour image segmentation, produces a non-negligible number of faulty measurements, it is better to use as a second measurement provider a robust detection module.

The other alternative is the use of the second provider, a method that is able to learn the target's appearance continuously. When the animal has a small size, it is preferable to use Boosting or CSRT. Boosting is more relevant for cases where a fast running time is required and the target is easily distinguishable from the background. CSRT can be used for a target with more complex appearances if a slower running time can be tolerated. MIL took longer than Boosting and CSRT alone, but it performed better than the other trackers implemented in version 4.1.0 of OpenCV contribution for small-scale animals. However, when MIL is used as the second measurement provider, the proposed VOT solution produces a non-negligible number of outliers. Therefore *Tracker*6 is not recommended. For medium-size animals, especially for handling occlusion with slow animals, the results show that the previous trend reverses with KCF, Median Flow,

GOTURN, and TLD the best choices. When using one of the trackers in OpenCV as the second measurement provider, the proposed tracking solution is more suitable for real-time applications than when using YOLOv3.

The SSIM index is not adapted to handle change detection in the case of small- scale animals. But it can be used to regularly change the two PF measurement providers. On the other hand, it is suitable for the handover, especially in presence of multiple particles either when considering the PF at the current time step or when generating new particles around the current PF's location estimate. Note that some recommendations were made for reducing the computation time required by the second and the third method for real-life implementation. Moreover, the handover communication protocol ensures the tracking continuation or stoppage and prevents there being more than two drones involved in a handover.

Chapter 6

Conclusion and Future Work

This project is about the development of a tracking system involving a formation of drones which collaborate in animal monitoring. The system is characterized by several components which are the key contributions of this study. These components are: the formation itself, the proposed VOT framework, the handover and the solution for the guidance of a drone-tracker based on the information available in the images captured by the drone-tracker during tracking. Several algorithms have being developed at the constituent-level for the system operation. The system primarily targets terrestrial animals.

The grid formation involves specific locations where the drones are placed. This ensures coverage of the field to monitor and the distribution of the tracking task amongst the drones. It also mitigates the impact of their energy level limitations. At a given time step, a drone is employed to track the target and to collect visual data. Furthermore, having such a distributed monitoring system can be beneficial for the respect of countries' regulations regarding foreign drone flights for parks and reserves which are shared between countries. Indeed, each country can have its drones flying only in its territories, but still contribute to a tracking using the handover which would happen at the common borders of these countries. Beyond animal monitoring applications, the proposed tracking system could also be adapted to the surveillance and the management of territories which are shared amongst countries.

The system uses as VOT solution the PF with two measurement providers which are decided at a time step based on the value of the SSIM index between the initial template for the target and an image patch centered at the anchor. The latter is the previous location estimated. However, the SSIM index is not adapted in this tracking framework when the target's scale is small. But, its use is not prohibited if one of the

two measurement providers is a reliable detection system or a good target appearance learning method. During the conception of the framework, the deep learning detection module selected was YOLO, more specifically YOLOv3 as it is one of the fastest object detectors with a good accuracy.

Yet, the tests have revealed that YOLOv3 is not adapted for the accurate detection of animals which are filmed by a drone flying at a high altitude. It might be due to the use of the ROI which is inevitable in the proposed tracking framework when a deep learning-based object detector is involved as second measurement furnisher. Indeed, since YOLOv3 was not explicitly trained on the animals, the ROI was employed to reduce the searching region where the target could be located. In addition, using the ROI helps to reduce YOLOv3's running time. Such reduction of YOLOv3's running time is important because the proposed tracking framework is intended to run in real-time. It is also possible that YOLOv3 fits the proposed framework if YOLOv3 is given the ability to accurately discriminate the target through training.

To overcome this shortcoming, tests were performed with some of the trackers implemented in OpenCV contribution version 4.1.0. These were primarily Boosting CSRT and MIL, for the MPI-AB dataset and KCF, Median Flow, GOTURN and TLD for the *cows video*. The tests show that the proposed tracking framework works when the second measurement provider can accurately differentiate the foreground from the background even with nearly random SSIM values. Note that the proposed tracker is slightly adapted to the second measurement provider when it is a method that learns the target's appearance. Indeed, such a measurement provider is required to run in parallel when the colour image segmentation furnishes measurements and it does not require the ROI.

The tests with *Tracker*6 which involves MIL as second measurement furnisher were inconclusive. *Tracker*6 was able to track the target for some tests while not for others. It seems it is because the tracking produced by MIL are not always precise with targets which have small resolution. Also, the PF's motion model involves noise and the colour image segmentation produces a considerable number of false positives. Indeed, the proposed VOT solution may produce outliers even with a reliable second measurement provider. Yet, much more tests is required to verify this hypothesis. On the other hand, the motion model's noise can also be investigated to assess its impact on the proposed tracking framework, especially for *Tracker*6.

The tests on the MPI-AB dataset shown that a tracking can be affected by the conditions in the field, in particular the daylight ambience. Indeed, most trackers which did not fail for the sequence $ob002 - 01a$ tracked the target's shadow. Image preprocessing techniques could be used to reduce the potential impact of some of the field conditions

on tracking.

The importance of accurately tracking a particular animal varies in function of the intended application. For example, if the aim of using the system in practice is only to monitor a group of animals of the same species, it would not matter that the proposed VOT solution (or any other suitable tracker) switches between animals of the same group (with similar colours and shapes). Note that if one were to use a specific motion model, this behaviour of the proposed VOT solution might not be desirable because the motion model might not be adapted to the new target. The drone-tracker will still be following the group of animals that the initial target belongs to. Therefore, YOLOv3 could be trained on the species instead of a singular animal while Boosting and CSRT would even be more adapted as they better discriminate a particular animal than YOLOv3. This is more realistic for long-term tracking.

The framework is suitable for real-time tracking. It reaches a speed of $28.7 fps$ for *Tracker*4 on a sequence from the MPI-AB dataset and $20.7 fps$ for *Tracker*8 on the *cows video* (on a Core-i7 computer with 2.2GHz frequency per core and 8GB of RAM). The proposed framework also presents other advantages. It can deal well with occasional missing data and wrong measurements due to the repetition of the previous measurement and to the PF which needs more than one time step to converge to a given location. It also has other features such as the motion model which could be further tuned as a basic one is utilized here. However, the framework also employs the anchor which is two-sided. When the tracking is accurate, the anchor is beneficial. But when it is not, the anchor can induce the failure of the whole tracking framework. It is why an accurate detection module is more relevant. Such a detection module can scan a more important and growing region (increase the ROI's size) near the anchor compared to learning trackers. Furthermore, the resampling threshold is not critical for the proposed VOT solution when it permits a minimal number of resampling. It should be because of the utilization of the anchor and the availability of two sources of measurements.

Following the realisation of certain conditions, the current drone relays its tracking to the most suitable drone. The handover relies on the communication protocol designed to ensure its success or the tracking termination in case of a failure. Simulations have been performed to show how the drones in a handover utilize the knowledge of their drone-partner for their next action. On the other hand, during a handover, the identity of the target can be validated with the value of the SSIM index between regions which are centered at the target's estimated location in the drones' *FOV*s. The drones'*FOV*s were simulated using frames available in the video sequences involved by this study. Indeed, data from two drone viewpoints were not available. This would be something

one could explore further, given suitable real-world data. The results obtained from the simulated drones' $FOVs$, have revealed that it is better to compare the patch in FOV_A to multiple patches in FOV_B, and to select the location of the patch that produces the highest SSIM value to update the target's position in FOV_B. In addition, it is better to use the transmitted particles when their number is high. It is typically the case for medium-size animals or to generate more new particles when the initially used number of particles is small, as can be observed with small-size animals. Note that the first method is not excluded for an animal viewed at a large scale.

Yet, in practice, for the second and the third method, a relatively high number of particles could delay the computation of the most likely position for the target in $DroneB$'s FOV. Several practical solutions were proposed to mitigate delays in handover induced by computations required by the second and the third methods. Amongst these recommendations are the increase of the spreading factor γ' for the particles when the third method is used and the parallelization of the calculations at this stage on both drones' hardware for the two methods. In addition, just after the success of the handover, the anchor would help the new drone-tracker to keep tracking the target. If a target's appearance learning technique is used as the second measurement furnisher, the new drone-tracker could also benefit from all of the information gathered by the former drone-trackers regarding the target's appearance which would be better discriminated.

The handover implies at a time step, the involvement of a drone-peer selected from the formation. Indeed, information about the handover frequency and the specific drones which are often involved, could provide insights about animals' behaviour. If for some tracks, the drones of a particular cell are often activated, it might be because the cell contains elements (for example, water points and tree types) which are appealing to the targets. On the contrary, if drones of a particular cell are not often used, it could be explained by the presence of threat or source of discomfort for the animals (for example, predator, poachers and lack of trees). For the system maintenance, drones which would be used less often could replace others which might be unavailable for a given reason.

The drone-tracker makes use of the outputs of the proposed tracking framework to follow the animal because there are no tracking devices on the target. These outputs are taken in between two time steps and are used to calculate the new yaw and pitch for the drone-tracker's position update. In between the two time steps, the drone-tracker hovers above the animal. The frequency at which a drone-tracker updates its position during a track, could advise on the target's state, i.e. if the latter is running away from a danger (such as a predator or a poacher), resting or harmed and unable to move.

As stated at the outset, monitoring of wildlife involving no contact, is important for

animal welfare, so methods such as non-intrusive aerial monitoring are sought. This study aims to make contributions in this area. The extent to which the methods developed met the goals are now summarized. The monitoring system relies on a grid formation which enables the distribution of the tracking amongst several drones. At a time step, a drone-tracker follows the target animal based only on what it films. A real-time tracking algorithm which makes use of the images captured by the drone-tracker is proposed to estimate the target's location. The method is based on the PF which uses two measurement providers which are switched based on the values of the SSIM index between the initial and the current appearance of the target. The outputs obtained from the proposed VOT solution are then used to compute the relevant yaw and pitch which are required to update the position of the drone-tracker. When the drone-tracker is running out of energy or when it arrives in a cell under the surveillance of other drones in the formation, it can select the most suitable one, to relay to the selected drone, the animal tracking via the handover. The handover exploits a communication protocol and a method to verify if the relevant animal is relayed from the drone-tracker to the drone-partner. These components together represent the contribution of the study for wildlife monitoring which is important for wildlife conservation. The proposed monitoring system is formulated as a state estimation problem distributed amongst recent technological development, drones, and involves mathematical techniques such as the PF, the SSIM index and machine learning techniques.

Future work could be to replace the colour segmentation technique by an approach better able to distinguish animals from backgrounds with a similar colour, to train YOLOv3 on the specific dataset and to test other versions of the SSIM index or to design a new image similarity evaluation technique. On the other hand, it could be interesting to employ a set of templates with several representations of the target to periodically update the single initial template used in this study. Moreover, one could allocate a subset of templates to a specific algorithm able to handle the VOT challenge represented by this subset to allow more than two measurement providers with the PF. A promising ensemble technique for tracking offline could involve the trackers which performed the best in this study in such a way that, at a time step, the ensemble prediction would be the median of these trackers' outputs.

Appendices

Appendix A

Algorithms

This appendix presents algorithms designed for this study and refer to in Chapter 4. These are algorithms for the implementation of the VOT solution (Algorithm 2), the handover between two drones (Algorithm 3), the selection of the drone-partner by the drone-peer (Algorithm 4), the actions of the two drones involved in the handover during communication (Algorithm 5 and 6) and the update of the drone-tracker's position (Algorithm 7). Each algorithm is associated with a table that contains the definition of the variables involved.

A.1 Algorithm for VOT Solution Implementation

A.2 Algorithm for Animal Tracking Handover between Two Drones

A.3 Algorithm for the Selection of the Drone-Partner by the Drone-Peer

A.4 Algorithms for the Actions of the Two Drones Involved in the Handover Communication Protocol

A.5 Algorithm for the Drone-Tracker's Position Update

Algorithm 2 Proposed VOT solution.

Require: The target's $template$ and the position in the first frame, F_{Width} and F_{Height}, the threshold for change detection $chg_{Tresh} \in [0,1]$ based on the SSIM index, the number of particles N.

Require: The $Area_{Tresh}$ to reduce the colour image segmentation candidates and $colour$ the colour threshold, the target's speed components v_x, v_y and the standard deviation std for the motion model, the number of time steps $anchor_{shift}$ where the $anchor$ (which is the previous PF estimate for the target's location) is equal to the measurements.

Require: The $yolo_{Weights}$, the $yolo_{Config}$, the $yolo_{Classes}$, the ROI's dimensions ROI_w, ROI_h, the dimensions $blob_x, blob_y$ to resize the ROI, the $yolo_{Confidence}$ for detections, the Non-maximum suppression method's confidence nms.

Ensure: The target's position estimate $\hat{x}_k(\hat{a}_k, \hat{b}_k)$ at each time step k.

Initialization of k to 1, $anchor$, $Area_{Thresh}$, N, std, chg_{Thresh}, $Meas_x$, $Meas_y$, BB, $yolo_{Net}$, $particles$, $particles_{Weights}$. ▷ $Meas_x$, $Meas_y$ are for storing measurements from the YOLO or the colour image segmentation.

while TRUE **do**

  Capture the current $frame$.

  if there is no new $frame$ **then**

    break

  end if

  Update all $particles$' positions using v_x, v_y, std. ▷ Equation 4.2.1

  $ROI_{ssim} = \left(anchor_x - \frac{template_x}{2}, anchor_y - \frac{template_y}{2}, template_x, template_y\right)$ ▷ Computation of BB's dimensions around $anchor$.

  Use ROI_{ssim} to obtain a $frame_{Crop}$.

  if $template_x > 3$ & $template_y > 3$ **then** ▷ SSIM index cannot be computed for images with dimensions smaller than 3×3.

    $ssim = SSIM(frame_{Crop}, template)$ ▷ Equation 4.2.7 and 4.2.14

  else

    $ssim = 0$

  end if

  $ssim = abs(ssim)$ ▷ Absolute value of $ssim$.

  if $ssim > chg_{Thresh}$ **then** ▷ Check if there is a potential change in the target's appearance using the SSIM.

    $ROI = \left(anchor_x - \frac{ROI_w}{2}, anchor_y - \frac{ROI_h}{2}, ROI_w, ROI_y\right)$

    Use ROI to extract $frame_{ROI}$ from $frame$.

    $frame_{ROI}$ preprocessing. ▷ Scaling and resizing to dimensions $(blob_x, blob_y)$.

    Get $outputs$ (predictions) from the three $yolo_{Net}$'s output layers. ▷ Figure 4.4

    Create empty lists: $classes, confidences, boxes, yolo_{Centers}, distances$.

    for det in $outputs$ **do**

      $scores = det[5]$

      $class_{Id} = \arg\max(scores)$

      $conf = scores[class_{Id}]$

      if $conf > yolo_{Confidence}$ **then**

        $center_x, center_y = det[0] * blob_x, det[1] * blob_y$ ▷ YOLO's BB outputs are normalized between [0,1].

Algorithm 2 Proposed VOT solution (cont...).

$w, h = det[2] * blob_x, det[3] * blob_y$

$x, y = center_x - \frac{w}{2}, center_y - \frac{h}{2}$

$boxes.append([x, y, w, h])$

$confidences.append(conf)$

$classes.append(class_{Id})$

$center_{xr} = center_x + anchor_x - \frac{ROI_x}{2}$ ▷ Find approximations of the coordinates in the video frames' coordinate system using their equivalents in the ROI's coordinate system.

$center_{yr} = center_y + anchor_y - \frac{ROI_y}{2}$

$yolo_{Centers}.append(center_{xr}, center_{yr})$

$ind_{nms} = NMS(boxes, confidences, yolo_{Conf}, nms)$ ▷ Use the Non-maximum suppression method to reduce candidates.

for each *centre* in $yolo_{Centers}$ **do**

$distances.append(\|anchor, centre\|)$ ▷ Equation 4.2.2

$index = \arg\min(distances)$

$index_{yolo} = ind_{nms}[index]$

$BB_x = boxes[index_{yolo}]_x + anchor_x - \frac{width}{2}$

$BB_y = boxes[index_{yolo}]_y + anchor_y - \frac{height}{2}$

$BB_w, BB_h = boxes[index_{yolo}]_w, boxes[index_{yolo}]_h$

$Meas_x, Meas_y = yolo_{Centers}[index_{yolo}][0], yolo_{Centers}[index_{yolo}][1]$

end for

end if

end for

else

$Meas_x, Meas_y, BB = imageSegmentation(frame, colour, anchor, Area_{Thresh})$ ▷ *colour* is the colour threshold (Equation 4.2.6) while *anchor* is used to select the best candidate segmented region (Equation 4.2.2).

end if

Update the $particles_{Weights}$ based on $Meas_x, Meas_y$.

Estimate $\hat{x}_k(\hat{a}_k, \hat{b}_k)$ from the *particles* and their *weights*. ▷ Equation 4.2.4

if the resampling condition is satisfied **then** ▷ Equation 4.2.5.

Resample the *particles*.

Attribute to all $particles_{Weights}$ the value $1/N$.

end if

if $k > anchor_{shift}$ **then** ▷ For the first time steps, the *anchor* is set to the first measurements for the PF convergence to the target's position.

$anchor = \hat{a}_k, \hat{b}_k$

else

$anchor = Meas_x, Meas_y$

end if

$k = k + 1$ ▷ This time step update is also valid for all of the other algorithms presented in this chapter since they can run in parallel.

end while

Table A.1: Definition of variables used in Algorithm 2. The following abbreviations are employed: colour image segmentation (CIS), region of interest (ROI).

Variables	**Definitions**
$anchor$	Previous estimate for target's position
$anchor_{shift}$	Number of initial time-steps where the $anchor$ is equal to the measurement
$Area_{Tresh}$	Area threshold to eliminate small regions segmented by CIS
$blob_x$	Width for YOLOv3's input
$blob_y$	Height for YOLOv3's input
BB	bounding box defined by top left corner coordinates and width and height
chg_{Tresh}	SSIM index threshold for change detection in the target's appearance
$colour$	Colour threshold
$frame$	Current video frame
F_{Width}	Video frame's width
F_{Height}	Video frame's height
$Meas_x$	Abscissa of measurement at time step k
$Meas_y$	Ordinate of measurement at time step k
nms	Confidence threshold for Non-maximum suppression
N	Number of particles
$outputs$	YOLOv3's predictions
$particles$	Set of particles
$particles_{Weights}$	Weights of particles
ROI_h	ROI's height
ROI_{ssim}	ROI centered at anchor for comparison with $template$ using SSIM index
ROI_w	ROI's width
$ssim$	Absolute value of SSIM index between target's initial and current appearance
std	Standard deviation of the zero-mean Gaussian white noise used in the PF's motion model
$template$	Image patch containing the initial appearance of the target
v_x	Constant speed along x-axis
v_y	Constant speed along y-axis
$yolo_{Classes}$	Classes for YOLOv3
$yolo_{Config}$	Configuration parameters of YOLOv3
$yolo_{Confidence}$	Confidence score threshold for YOLOv3's detections
$yolo_{Net}$	YOLOv3 network (three outputs layers)
$yolo_{Weights}$	Weights from trained YOLOv3

Algorithm 3 Handover.

Require: The time step k (which is updated in Algorithm 2), the threshold E_{Thresh} for the drone energy level, the maximal number of attempts $n_A \geq 1$ if the target identification fails, the maximal number of drones $n_D \geq 1$ if $DroneA$ needs to call another $DroneB$ candidate, The minimum acceptable SSIM value for the ROIs comparison $ssim_{Thresh} \in [0, 1]$.

Require: The target's $template$ from the first frame, the threshold for change detection $chg_{Tresh} \in [0, 1]$ (used in Algorithm 2. Yet, another threshold could be set).

Require: The number of particles N, the particles and their weights $\{(x_k^i, w_k^i)|i = 1, 2, \cdots, N\}$, the corresponding target's position estimate $\hat{x}_k$ (in FOV_A), the distances δ_x and δ_y for shifting the transmitted particles' positions in FOV_B.

Ensure: $success$ is $TRUE$ or $n_{TA} > n_A$.

Initialization: $success = FALSE$, $ctr_D = 1$.
while $Energy_{DroneA} \geq E_{Thresh}$ & $success = FALSE$ **do**
  if $ctr_D \leq n_D$ **then**
    $DroneA$ selects a new $DroneB$. ▷ Algorithm 4
    $ctr_D = ctr_D + 1$
    $n_{TA} = 0$
  else
    Go to 30 ▷ Continue the tracking up to the allocated energy level termination.
  end if
  while $n_{TA} \leq n_A$ & $success = FALSE$ **do**
    Get the current $\hat{x}_k$, $\{(x_k^i, w_k^i)|i = 1, 2, \cdots, N\}$.
    Extract the ROI_A around $\hat{x}_k$ in FOV_A.
    if $|ssim(ROI_A, template)| \geq chg_{Tresh}$ **then** ▷ Check if $DroneA$ is still locked on the initial target.
      Use δ_x, δ_y, $\hat{x}_k$ and $\{(x_k^i, w_k^i)|i = 1, 2, \cdots, N\}$ to extract ROI_B from FOV_B.
      if $|ssim(ROI_A, ROI_B)| \geq ssim_{Thresh}$ **then** ▷ The target is successfully identified by $DroneB$.
        $DroneB$ and $DroneA$ acknowledge each others. ▷ Section 4.3.3 gives further details in the opposite case.
        $DroneA$ enters the mode $Return$.
        $DroneB$ becomes the new $DroneA$.
        $DroneA$ enters the mode $Track$.
        $success = TRUE$
      end if
      $n_{TA} = n_{TA} + 1$
    else
      $n_{TA} = n_{TA} + 1$
    end if
  end while
  if $success = FALSE$ **then**
    $DroneB$ enters the mode $Return$.
  end if
  if $Energy_{DroneA} < E_{Thresh}$ **then**
    $DroneA$ quits the mode $Track$.
    $DroneA$ enters the mode $Return$.
  end if
end while

Table A.2: Definition of variables used in Algorithm 3. The following abbreviations are employed: region of interest (ROI), field of view (FOV).

Variables	**Definitions**
chg_{Tresh}	SSIM index threshold for change detection in the target's appearance
ctr_D	Counter of number of successive *DroneB* candidates
$Energy_{DroneA}$	Current battery life for *DroneA*
E_{Thresh}	Threshold for *DroneA*'s battery life
n_A	Number of attempts
n_D	Maximal number of drones that *DroneA* can try to handover a tracking to if attempts fail successively
n_{TA}	Counter for *DroneA*'s attempts to handover a tracking
N	Number of particles
ROI_A	ROI in *DroneA*'s FOV
ROI_B	ROI in *DroneB*'s FOV
$ssim_{Thresh}$	Threshold for SSIM index to confirm (or not) the target's identity
$success$	Boolean variable for the handover state (successful or unsuccessful)
$template$	Image patch containing the initial appearance of the target
δ_x	Value to shift particles transmitted by *DroneA* to *DroneB* in x direction
δ_y	Value to shift particles transmitted by *DroneB* to *DroneB* in y direction

Table A.3: Definition of variables used in Algorithm 4.

Variables	**Definitions**
$Join$	Boolean variable for the two drones' alignment (successful or unsuccessful)
k_{Join}	Interval of time steps for checking on the alignment of *DroneA* and *DroneB* for a handover
L_B	Set of drones for the current cell
${}^{W}_{A}X_k$	*DroneA*'s position in the real world
${}^{W}_{B}X_k$	*DroneB*'s position in the real world

Algorithm 4 Peer-drone selection.

Require: The time step k (which is updated in Algorithm 2), the periodicity k_{Join} for checking on both drones alignment for the handover. ▷ Alignment in Figure 4.5.

Ensure: The best candidate *DroneB* selection and displacement near *DroneA*.

1: *DroneA* activates its GPS (if it was not activated). ▷ It can be another global positioning system.
2: Get *DroneA*'s current position ${}^{W}_{A}X_k$ in the real world.
3: Use ${}^{W}_{A}X_k$ to identify the current cell.
4: From the current cell get the set L_B ($\#L_B < 5$) of *DroneB* candidates. ▷ These are the drones in *Sleep* modes which are responsible for the current cell. L_B should contain at least 1 element.
5: **if** $\#L_B > 1$ **then**
6: Remove from L_B the drone candidate with the lowest energy level.
7: **end if**
8: **if** $\#L_B > 1$ **then**
9: Remove from L_B the farthest drone from *DroneA*.
10: **end if**
11: **if** $\#L_B > 1$ **then**
12: Remove from L_B the drone which has the lowest disk space.
13: **end if**
14: Select as *DroneB* the remaining drone in L_B.
15: *DroneB* changes its *Sleep* state to *Handover* state.
16: *DroneB* activates its GPS.
17: $Join = FALSE$
18: **while** $Join = FALSE$ **do**
19: **if** $k \ mod \ k_{Join} = 0$ **then**
20: Get *DroneA*'s current position ${}^{W}_{A}X_k$.
21: Get *DroneB*'s current position ${}^{W}_{B}X_k$.
22: **if** ${}^{W}_{A}X_k$ similar to ${}^{W}_{B}X_k$ **then** ▷ If both drones are near to each other (Figure 4.5).
23: $Join = TRUE$
24: **else**
25: *DroneB* continues flying to join *DroneA*.
26: **end if**
27: **end if**
28: **end while**

Algorithm 5 *DroneA*'s actions during a handover communication protocol.

Require: The time step k (which is updated in Algorithm 2), $\beta \geq 1$, $n_A \geq 1$ and $n_D \geq 1$, $\zeta = 1/\zeta', \zeta' \in \mathbb{N}$.
Ensure: *DroneA* takes at least one action at each time step k'.

Initialization: $n_{TA} = 0$, $w_A = 0$, $k' = k$.
if there is a new message **then**
    Receive the message.
    $n_{TA} = 0$
    $wait_A = 0$
end if
if it is the second action **then**
    Send $K_A\rho$.
    $n_{TA} = n_{TA} + 1$
    if there is no variable ctr_D **then**
        Create temporary variable ctr_D.
        $ctr_D = 1$
    end if
else if the previous action was to receive a message **then**
    Send $K_A K_B K_A \rho$.
    Destroy temporary variable ctr_D.
    Enter the mode *Return*.
else if the previous action was to send or to wait for a message **then**
    if $w_A < \beta - 1$ **then**
        wait for message $K_B k_A \rho$.
        $w_A = w_A + 1$
    else
        if $n_{TA} < n_A$ **then**
            $w_A = 0$
            Send $K_A\rho$.
            $n_{TA} = n_{TA} + 1$
        else
            if $ctr_D \leq n_D$ **then**
                Select and call a new *DroneB*.
                $ctr_D = ctr_D + 1$
                $n_{TA} = 0$
                $w_A = 0$
            else
                Continue *Track*.
            end if
        end if
    end if
else
    Pass
end if
$k' = k' + \zeta$ ▷ $\zeta = \dfrac{1}{\zeta'}, \zeta' \in \mathbb{N}$. ζ is used to discretize the time step k.

Algorithm 6 *DroneB*'s actions during a handover communication protocol.

Require: The time step k (which is updated in Algorithm 2), $\beta \geq 1$, $n_B \geq 1$, $n_A \geq 1$,$\zeta = 1/\zeta', \zeta' \in \mathbb{N}$.
Ensure: *DroneB* takes at least one action at each time step k'.

Initialization: $n_{TB} = 0$, $w_B = 0$, $lev = 1 (lev \in \{1, 2, 3\})$.
if there is a new message **then**
    Receive the message.
    $n_{TB} = 0$
    $w_B = 0$
    if message is $K_A\rho$ **then**
        $lev = 2$
    else
        $lev = 3$
    end if
end if
if it is the second action **then**
    wait for the first message $k_A\rho$.
    $w_B = w_B + 1$
else if the previous action was to receive a message **then**
    if $lev = 2$ **then**
        Send $K_B K_A \rho$.
        $n_{TB} = n_{TB} + 1$
    else
        Enter the mode *Track*.
    end if
else if the previous action was to send or to wait for a message **then**
    if $lev = 1$ **then**
        if $w_B < n_A(\beta + 1)$ **then**
            wait for a message $K_A\rho$.
            $w_B = w_B + 1$
        else
            Enter the mode *Return*.
        end if
    else if $lev = 2$ **then**
        if $w_B < \beta$ **then**
            wait for a message.
            $w_B = w_B + 1$
        else
            if $n_{TB} < n_B$ **then**
                Send $K_B K_A \rho$.
                $n_{TB} = n_{TB} + 1$
                $w_B = 0$
            else
                Enter the mode *Return*.
            end if
        end if
    else
        Enter the mode *Return*.
    end if
else
    Pass
end if
$k' = k' + \zeta$ ▷ Only if not completed previously by *DroneA*.

Algorithm 7 Drone-tracker (*DroneA*) visual guidance.

Require: The time step k (which is updated in Algorithm 2), constants used in the equations involved in this algorithm, i.e. F_{Height}, F_{Width}, v_{Max}, γ, Φ_{Comp} and Θ_{Comp}.
Require: The maximum number of time steps k_{Stop}, the minimal energy level E_{Stop} to return home and the stopping-cell's boundaries GPS coordinates $Cell_{Stop}$.
Ensure: *DroneA*'s position update and global tracking algorithm stoppage.

DroneA activates its GPS (if it was not activated).
while TRUE **do** ▷ While *DroneA* is in the *Track* mode.
  Get $\hat{x}_k(\hat{a}_k, \hat{b}_k)$ from the proposed tracking framework. ▷ Algorithm 2
  Calculate and store ${}^{F}\hat{x}_k({}^{F}\hat{a}_k, {}^{F}\hat{b}_k)$ from $\hat{x}_k(\hat{a}_k, \hat{b}_k)$. ▷ Equations 4.4.1 and 4.4.2
  Compute κ_{Max}. ▷ Equation 4.4.4
  if it is a handover process **then**
    $\kappa = \kappa_{Handover}$ ▷ $\kappa_{Handover} = 1, 2, \cdots, \frac{\kappa_{Max}}{2}$.
  else
    $\kappa = \kappa_{Max}$
  end if
  if $k \; mod \; \kappa = 0$ **then**
    Get $\hat{x}_k(\hat{a}_k, \hat{b}_k)$.
    Compute ${}^{F}\hat{x}_k({}^{F}\hat{a}_k, {}^{F}\hat{b}_k)$ from $\hat{x}_k(\hat{a}_k, \hat{b}_k)$.
    Find ${}^{F}\phi_k, {}^{F}\theta_k$ using ${}^{F}\hat{x}_{k-\kappa}$ and ${}^{F}\hat{x}_k$. ▷ Equations 4.4.5, 4.4.6 and 4.4.7
    Calculate ${}^{W}\Phi_k, {}^{W}\Theta_k$ from ${}^{F}\phi_k, {}^{F}\theta_k$. ▷ Equation 4.4.8
    Update *DroneA*'s position using ${}^{W}\Phi_k, {}^{W}\Theta_k$.
  else
    DroneA hovers.
  end if
  Get *DroneA*'s current position ${}^{W}_{A}X_k$ in the real world.
  if $k > k_{Stop}$ OR $Energy_{DroneA} < E_{Stop}$ OR (${}^{W}_{A}X_k$ within $Cell_{Stop}$) **then**
    Enter the mode *Return*.
  end if
end while

Table A.4: Definition of variables used in Algorithm 5.

Variables	**Definitions**
ctr_D	Counter of number of successive *DroneB* candidates
k'	Time step in handover communication protocol
n_A	Maximal number of attempts for message sending by *DroneA*
n_D	Maximal number of drones that *DroneA* can try to handover a tracking to if attempts fail successively
n_{TA}	*DroneA*'s messages counter at a given level of communication (amongst the three levels of communication)
w_A	Counter for a message waiting time
β	Variable to define size of the time-receptive window of a message
ζ	Discrete value to increment time step k'

Table A.5: Definition of variables used in Algorithm 6.

Variables	**Definitions**
k'	Time step in handover communication protocol
lev	Current level of communication
n_B	Maximal number of attempts for message sending by *DroneB*
n_{TB}	*DroneB*'s messages counter at a given level of communication (amongst the three levels of communication)
w_B	Counter for a message waiting time
β	Variable to define size of the time-receptive window of a message
ζ	Discrete value to increment time step k'

Table A.6: Definition of variables used in Algorithm 7. The symbol * indicates that the variable does not appear explicitly in Algorithm 7 and is required for the computation of another variable.

Variables	**Definitions**
$Cell_{Stop}$	Coordinates of limits of the cell where tracking instance could be stopped
E_{Stop}	Minimal battery life for *DroneA* to stop tracking
F_{Height}	Frame height
F_{Width}	Frame width
k_{Stop}	Total duration of tracking
v_{Max}	Animal's maximal speed in video frame
${}^{F}\hat{x}_k({}^{F}\hat{a}_k, {}^{F}\hat{b}_k)$	Outputs of VOT solution converted in a given Cartesian coordinate system in video frame
${}^{W}_{A}X_k$	*DroneA*'s current position in the real world
γ*	Zero-mean Gaussian white noise
κ	Number of frames between two updates of *DroneA*'s position
$\kappa_{Handover}$	Maximal number of frames between two updates of *DroneA*'s position during a handover process
κ_{Max}	Maximal number of frames between two updates of *DroneA*'s position
${}^{F}\theta_k$	Angle by which the target rotated in video frames during *kappa*
Θ_{Comp}*	Constant value to add to the updated value for *DroneA*'s yaw
${}^{W}\Theta_k$	Yaw for *DroneA*'s position update
${}^{F}\phi_k$	Distance travelled by the target in video frames during *kappa*
${}^{W}\Phi_k$	Pitch for *DroneA*'s position update
Φ_{Comp}*	Constant value to add to the updated value for *DroneA*'s pitch

Appendix B

Extension of the Case Study for the VOT Solution

This appendix presents more details about the application of the proposed VOT solution to the *cows video*. As mentioned in Subsection 5.1.1, this video is available in the *deepgaze* library. The video presents a challenging occlusion situation. As a reminder, this video was used for testing and to illustrate features of the proposed VOT solution, although it uses a different setup than that intended for the application.

B.1 Manual Count of Images (without Occlusion) for the Trackers Outputs

To quantitatively evaluate the trackers' performance on the *cows video* in particular on handling the occlusion situation, the number of frames where each tracker misses detections after the beginning of the occlusion was manually determined. The first frames are not used because the tracked animal barely moves as it is grazing. Therefore, all the trackers tracked the targeted cow up to the beginning of the occlusion, even if TLD was given from time to time false positives due to its detection module and *Tracker*3 tracked another cow in some of the first frames.

A tracker's estimate was considered as a detection if it falls on the initially tracked animal's body. For this reason, all of the trackers have a minimal number of missed detections since the target was fully occluded at some point by the passing cow. Also, an estimate located outside and near the contour of the target was considered as a detection, if this estimate fell inside a bounding box that partially contains the target. These estimates have to be considered because they are provided by trackers that produce

bounding boxes which also contain the background.

To mitigate the human-error factor in this process, a specific methodology was employed. For each tracker, it was first produced and represent on the entire *cows video*, the averaged of the results for six runs. From the latter were extracted the occlusion's sequence which starts around the 522^{th} frame. Next, a folder was created for each tracker. A tracker's folder gathered images obtained by cropping all of the frames of its video sequence's outputs to have a better view of the occlusion compared to if whole frames were considered. The content of each folder was browsed to visually identified images where the target was not detected. Next, the images identified were placed in a subfolder. The contents of each folder and subfolder were re-evaluated to identify missed classifications which were corrected when found. The final stage was to determine the number of successful detections. This number corresponds to the length of a tracker's folder minus one (for the subfolder).

A similar process was employed to find the number of frames without occlusion in the sub video sequence. Here it was considered, the dual of the frames where the passing cow seemed to touch the target, up to the frame where both cows separated. A total of 136 frames was found where there was no occlusion. Since the sub video sequence has a length of 459 frames, the ratio of frames with no occlusions is 0.296. This number was utilized as a threshold to identify the trackers which were able to handle the occlusion. Indeed, all the trackers with their ratio above this value are the successful trackers.

B.2 Tests and Interpretation of the Results for Tracking the Passing Cow

some tests were performed with the passing cow to better understand how the proposed tracking framework works. The results are briefly interpreted. MOSSE and KCF have shifted the tracking to the static cow after the occlusion. It seems, their CFs were not able to discriminate between the two cows here. Median-Flow and GOTURN started tracking other cows in the scene, even before the occlusion while TLD was detecting many false positives. Boosting, CSRT and MIL handled well the occlusion although CSRT was more confused since its bounding box became larger after the occlusion. The three of them started tracking the background when the target was out of view. Apart from MOSSE which did not perform well here also, the trend has reversed for the trackers. The trackers which were able to handle the occlusion when the static cow was the target are not able to here. Some of these trackers failed in tracking even before the occlusion

occurred. The trackers which failed when the static cow was the target, succeeded for the passing cow. Therefore, the trackers are sensitive to the target's motion and the type of occlusion.

Similar to Boosting, CSRT, and MIL, *Tracker*$1, 2a, 2b$ were able to follow the passing cow up to the moment it went out of view. Figure B.1 shows how *Tracker*$2a$ handled the occlusion. Indeed, the fact that the SSIM value was above the set threshold of 0.6 during the occlusion, has only permitted the colour image segmentation to furnish the measurements. The anchor helped here to eliminate the false positives generated by the colour image segmentation. It also explains why *Tracker*1 was successful. *Tracker*3 failed and tracked another cow from the beginning of the video.

Figure B.1: Example of *Tracker*$2a$'s tracking for the passing cow in the *cows video* for an SSIM threshold of 0.6. Frames are displayed from left to right and from top to bottom starting with the 518^{th} frame with interval 13 frames. Above each frame is the corresponding SSIM value. The SSIM values were all above the set threshold. Due to these values, only the colour image segmentation furnished measurements that were reliable.

*Tracker*4 tracked the passing cow similarly to Boosting. *Tracker*$5, 6$ tracked up to after the occlusion. However, they ended up losing the passing cow before the latter started exiting the frames. Around the 838^{th} and 839^{th} frames, the SSIM index value slightly dropped allowing CSRT and Boosting to provide the measurements. A change

was also detected by the SSIM index for the same frames in some of the videos in the averaged results of *Tracker*2*a* (the presence of the ROIs and the bounding boxes). Since it was only two measurements all the three trackers continued tracking the passing cow. Yet they were weakened. Their estimates were closer to the animal boundaries than to its center. Note that prior to the 838^{th} frame, the measurements furnished by the colour segmentation were also weak because the latter was segmenting the two cows like the case that is presented in Figure 5.16.

At the 842^{th} frame, CSRT provided almost all of the remaining measurements for *Tacker*5 and they were faulty. For *Tracker*6, MIL has misled the colour image segmentation which has continued to provide weak measurements from the 840^{th} frame. MIL took over at the 853^{th} frame and both gave wrong measurements. This shows that when the second measurement provider gives wrong measurements, it can negatively impact the colour segmentation and further lead to the loss of the target. It is not clear why MIL was unable to accurately represent the target's appearance after the occlusion as it did when it was alone since it can also learn independently in *Tracker*6 the target's appearance. But it might be because MIL involves randomness which leads to different tracks for different executions. For *Tracker*2*a*, after the 840^{th} frame, the colour segmentation was able to accurately segment the passing cow. This confirms the influence of MIL on the colour segmentation in *Tracker*6 and the need for at least one reliable provider.

A smaller threshold of value 0.5 is employed with *Tracker*5 and 6. The slight change detected at the 838^{th} frame when the SSIM index was 0.6 was no longer detected. The colour image segmentation completed successfully the tracking of the passing cow for *Tracker*5 and 6, but with a longer running time (compared to *Tracker*1, 2*a*, 2*b*, CSRT, and MIL) since CSRT and MIL were learning in parallel. Once again, it evidences the need for a reliable similarity metric, in particular, one which is less data-dependent.

List of References

[1] Gerstein, J. and Hernandez, D.: 20 extinct animals we've lost in the past 150 years. https://www.popularmechanics.com/science/animals/g201/recently-extinct-animals-list-470209/?slide=20. Aug 27, 2019. Accessed on: Feb 9, 2020.

[2] Harvey, R.J., Roskilly, K., Buse, C., Evans, H.K., Hubel, T.Y. and Wilson, A.M.: Determining position, velocity and acceleration of free-ranging animals with a low-cost Unmanned Aerial System. *Journal of Experimental Biology*, vol. 219, no. 17, pp. 2687–2692, 2016.

[3] Brooks, C., Bonyongo, C. and Harris, S.: Effects of global positioning system collar weight on zebra behavior and location error. *The Journal of Wildlife Management*, vol. 72, no. 2, pp. 527–534, 2008.

[4] Rasiulis, A.L., Festa-Bianchet, M., Couturier, S. and Côté, S.D.: The effect of radio-collar weight on survival of migratory caribou. *The Journal of Wildlife Management*, vol. 78, no. 5, pp. 953–956, 2014.

[5] Bennitt, E., Bartlam-Brooks, H.L., Hubel, T.Y. and Wilson, A.M.: Terrestrial mammalian wildlife responses to Unmanned Aerial Systems approaches. *Scientific Reports*, vol. 9, no. 1, pp. 1–10, 2019.

[6] Vas, E., Lescroël, A., Duriez, O., Boguszewski, G. and Grémillet, D.: Approaching birds with drones: first experiments and ethical guidelines. *Biology Letters*, vol. 11, no. 2, p. 20140754, 2015.

[7] Koh, L.P. and Wich, S.A.: Dawn of drone ecology: low-cost autonomous aerial vehicles for conservation. *Tropical Conservation Science*, vol. 5, no. 2, pp. 121–132, 2012.

[8] Zhong, W., Lu, H. and Yang, M.-H.: Robust object tracking via sparse collaborative appearance model. *IEEE Transactions on Image Processing*, vol. 23, no. 5, pp. 2356–2368, 2014.

[9] El-Shafie, A.-H.A., Zaki, M. and Habib, S.E.-D.: Fast CNN-based object tracking using localization layers and deep features interpolation. In: *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pp. 1476–1481. IEEE, 2019.

[10] Dalal, N. and Triggs, B.: Histograms of oriented gradients for human detection. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893. IEEE, 2005.

[11] Nam, H. and Han, B.: Learning multi-domain convolutional neural networks for visual tracking. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4293–4302. 2016.

[12] Held, D., Thrun, S. and Savarese, S.: Learning to track at 100 fps with deep regression networks. In: *European Conference on Computer Vision*, pp. 749–765. Springer, 2016.

[13] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R. and LeCun, Y.: Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[14] Girshick, R., Donahue, J., Darrell, T. and Malik, J.: Region-based convolutional networks for accurate object detection and segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 1, pp. 142–158, 2015.

[15] Redmon, J. and Farhadi, A.: YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[16] Wang, Z., Bovik, A.C., Sheikh, H.R. and Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[17] Gordon, N.J., Salmond, D.J. and Smith, A.F.M.: Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F Radar and Signal Processing*, vol. 140, no. 2, pp. 107–113, 1993.

[18] Biswas, D., Su, H., Wang, C. and Stevanovic, A.: Speed estimation of multiple moving objects from a moving UAV platform. *ISPRS International Journal of Geo-Information*, vol. 8, no. 6, p. 259, 2019.

[19] Gupta, S.G., Ghonge, D., Jawandhiya, P.M. *et al.*: Review of Unmanned Aircraft System (UAS). *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume*, vol. 2, 2013.

[20] Mogili, U.R. and Deepak, B.: Review on application of drone systems in precision agriculture. *Procedia Computer Science*, vol. 133, pp. 502–509, 2018.

[21] Hildmann, H. and Kovacs, E.: Using Unmanned Aerial Vehicles (UAVs) as Mobile Sensing Platforms (MSPs) for disaster response, civil security and public safety. *Drones*, vol. 3, no. 3, p. 59, 2019.

[22] Hassanalian, M. and Abdelkefi, A.: Classifications, applications, and design challenges of drones: A review. *Progress in Aerospace Sciences*, vol. 91, pp. 99–131, 2017.

[23] Shakhatreh, H., Sawalmeh, A.H., Al-Fuqaha, A., Dou, Z., Almaita, E., Khalil, I., Othman, N.S., Khreishah, A. and Guizani, M.: Unmanned Aerial Vehicles (UAVs): A survey on civil applications and key research challenges. *IEEE Access*, vol. 7, pp. 48572–48634, 2019.

[24] Brooke-Holland, L.: Unmanned Aerial Vehicles (drones): an introduction. *House of Commons Library, UK*, 2012.

[25] Arjomandi, M., Agostino, S., Mammone, M., Nelson, M. and Zhou, T.: Classification of Unmanned Aerial Vehicles. *Report for Mechanical Engineering class, University of Adelaide, Adelaide, Australia*, 2006.

[26] Weibel, R. and Hansman, R.J.: Safety considerations for operation of different classes of UAVs in the NAS. In: *AIAA 4th Aviation Technology, Integration and Operations (ATIO) Forum*, p. 6244. 2004.

[27] Wikipedia: Boeing condor. https://en.wikipedia.org/wiki/Boeing_Condor. Sept 28, 2019. Accessed on: Apr 26, 2020.

[28] Hassanalian, M., Abdelkefi, A., Wei, M. and Ziaei-Rad, S.: A novel methodology for wing sizing of bio-inspired flapping wing micro air vehicles: theory and prototype. *Acta Mechanica*, vol. 228, no. 3, pp. 1097–1113, 2017.

[29] Kahn, J.M., Katz, R.H. and Pister, K.S.: Next century challenges: mobile networking for "Smart Dust". In: *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 271–278. 1999.

[30] Austin, R.: *Unmanned Aircraft Systems: UAV design, development and deployment*, vol. 54. John Wiley & Sons, 2011.

[31] Ubaya, H. and Iqbal, M.: First person view on flying robot for real time monitoring. *ICON-CSE*, vol. 1, no. 1, pp. 41–44, 2015.

[32] O'Connor, R.: Developing a multirotor UAV platform to carry out research into autonomous behaviours, using on-board image processing techniques. *BE Thesis, Faculty of Engineering, Computing and Mathematics, University of Western Australia*, 2013.

[33] Archeology, D.: Anatomy of a UAV. http://dronearchaeology.com/portfolio_page/anatomy-of-a-uav/. Accessed on: May 2, 2020.

[34] Coach, U.: How to fly a drone a beginner's guide to multirotor systems & flight proficiency. https://uavcoach.com/how-to-fly-a-quadcopter-guide/#guide-4. Accessed on: May 2, 2020.

[35] Šustek, M. and Úředníček, Z.: The basics of quadcopter anatomy. In: *MATEC Web of Conferences*, vol. 210, p. 01001. EDP Sciences, 2018.

[36] McAndrew, I.R., Navarro, E. and Witcher, K.: Propeller design requirements for quadcopters utilizing variable pitch propellers. *International Journal of Materials, Mechanics and Manufacturing*, vol. 6, no. 1, p. 51, 2018.

[37] Omega, D.: The beginner's guide to drone motor essentials. https://www.droneomega.com/drone-motor-essentials/, . Accessed on: May 2, 2020.

[38] van Duijn, L.: Drone anatomy 101. https://vdr.one/drone-anatomy-101/. Sept 11, 2016. Accessed on: May 3, 2020.

[39] Chen, S., F Laefer, D. and Mangina, E.: State of technology review of civilian UAVs. *Recent Patents on Engineering*, vol. 10, no. 3, pp. 160–174, 2016.

[40] Corrigan, F.: Drone gyro stabilization, IMU and flight controllers explained. https://www.dronezon.com/learn-about-drones-quadcopters/three-and-six-axis-gyro-stabilized-drones/, . July 2, 2019. Accessed on: May 3, 2020.

[41] Flynt, J.: What sensors do drones use? https://3dinsider.com/drone-sensors/. Apr 18, 2019. Accessed on: May 4, 2020.

[42] Corrigan, F.: How do drones work and what is drone technology. https://www.dronezon.com/learn-about-drones-quadcopters/what-is-drone-technology-or-how-does-drone-technology-work/, . Apr 17, 2020. Accessed on: May 3, 2020.

[43] Omega, D.: How GPS drone navigation works. https://www.droneomega.com/gps-drone-navigation-works/, . Accessed on: May 4, 2020.

[44] Omega, D.: The advanced quadcopter battery guide. https://www.droneomega.com/quadcopter-battery-guide/, . Accessed on: May 4, 2020.

[45] Corrigan, F.: 12 best follow me drones and follow you technology reviewed. https://www.dronezon.com/drone-reviews/best-follow-me-gps-mode-drone-technology-reviewed/, . Nov 24, 2019. Accessed on: May 4, 2020.

[46] Matt Swider, M.W.: The best drone 2020: DJI, parrot and more for beginners and pros. https://www.techradar.com/news/best-drones. Apr 29, 2020. Accessed on: May 4, 2020.

[47] Linux Foundation, T.: Linux Foundation and leading technology companies launch open source Dronecode Project. https://www.linuxfoundation.org/press-release/2014/10/linux-foundation-and-leading-technology-companies-launch-open-source-dronecode-project/. Oct 13, 2014. Accessed on: May 4, 2020.

[48] Ardupilot. https://ardupilot.org/. Accessed on: May 4, 2020.

[49] MultiWii: http://www.multiwii.com/. Accessed on: May 5, 2020.

[50] LibrePilot: https://www.librepilot.org/site/index.html. Accessed on: May 5, 2020.

[51] AutoQuad: http://autoquad.org/. Accessed on: May 5, 2020.

[52] Foundation, D.: https://www.dronecode.org/. Accessed on: May 5, 2020.

[53] UAV, P.: Welcome to paparazzi UAV. https://wiki.paparazziuav.org/wiki/Main_Page. Accessed on: May 5, 2020.

[54] (ROS), R.O.S.: About ROS. https://www.ros.org/about-ros/. Accessed on: May 5, 2020.

[55] Autopilot, P.: https://px4.io/. Accessed on: May 5, 2020.

[56] Rao, B., Gopi, A.G. and Maione, R.: The societal impact of commercial drones. *Technology in Society*, vol. 45, pp. 83–90, 2016.

[57] Corrigan, F.: How to secure your drone from hackers permanently. https://www.dronezon.com/learn-about-drones-quadcopters/how-to-protect-your-drone-from-hackers-permanently/, . Nov 14, 2020. Accessed on: May 3, 2020.

[58] Mulero-Pázmány, M., Jenni-Eiermann, S., Strebel, N., Sattler, T., Negro, J.J. and Tablado, Z.: Unmanned Aircraft Systems as a new source of disturbance for wildlife: A systematic review. *PloS one*, vol. 12, no. 6, p. e0178448, 2017.

[59] EnergyOR: EnergyOR demonstrates multirotor UAV flight of 3 hours, 43 minutes. http://energyor.com/news/post:30. June 12, 2015. Accessed on: May 4, 2020.

[60] Energy, I.: Metavista breaks world record of multi rotor UAV flight time using intelligent energy fuel cell power module. https://www.intelligent-energy.com/news-and-events/company-news/2019/04/16/metavista-breaks-guinness-world-record-of-multi-rotor-uav-flight-time-using-intelligent-energy-fuel-cell-power-module/. Apr 16, 2019. Accessed on: July 2, 2020.

[61] Skysense: Autonomous charging in outdoor environment. https://www.skysense.co/charging-pad-outdoor. Accessed on: July 2, 2020.

[62] Authority, C.A.: Cap 722 Unmanned Aircraft System operations in UK airspace-guidance. *Directorate of Airspace Policy*, 2010.

[63] Fiaz, M., Mahmood, A. and Jung, S.K.: Tracking noisy targets: A review of recent object tracking approaches. *arXiv preprint arXiv:1802.03098*, 2018.

[64] Yilmaz, A., Javed, O. and Shah, M.: Object tracking: A survey. *ACM Computing Surveys (CSUR)*, vol. 38, no. 4, pp. 13–es, 2006.

[65] Canny, J.: A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[66] Horn, B.K. and Schunck, B.G.: Determining optical flow. In: *Techniques and Applications of Image Understanding*, vol. 281, pp. 319–331. International Society for Optics and Photonics, 1981.

[67] Lucas, B.D. and Kanade, T.: An iterative image registration technique with an application to stereo vision. In: *IJCAI'81 Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*, pp. 674–679. 1981.

[68] Harris, C.G. and Stephens, M.: A combined corner and edge detector. In: *Proc. 4th Alvey Vision Conf., Manchester, U.K., Aug. 1988*, pp. 147–151. 1988.

[69] Moravec, H.P.: Visual mapping by a robot rover. In: *IJCAI'79 Proceedings of the 6th international joint conference on Artificial intelligence - Volume 1*, pp. 598–600. 1979.

[70] Shi, J. and Tomasi, C.: Good features to track. In: *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 593–600. IEEE, 1994.

[71] Lowe, D.G.: Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[72] Comaniciu, D. and Meer, P.: Mean shift analysis and applications. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, pp. 1197–1203. IEEE, 1999.

[73] Wu, Z. and Leahy, R.: An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 11, pp. 1101–1113, 1993.

[74] Shi, J. and Malik, J.: Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[75] Wren, C.R., Azarbayejani, A., Darrell, T. and Pentland, A.P.: Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 780–785, 1997.

[76] Li, L. and Leung, M.K.: Integrating intensity and texture differences for robust change detection. *IEEE Transactions on Image Processing*, vol. 11, no. 2, pp. 105–112, 2002.

[77] Rittscher, J., Kato, J., Joga, S. and Blake, A.: A probabilistic background model for tracking. In: *European Conference on Computer Vision*, pp. 336–350. Springer, 2000.

[78] Rowley, H.A., Baluja, S. and Kanade, T.: Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 23–38, 1998.

[79] Viola, P., Jones, M.J. and Snow, D.: Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision*, vol. 63, no. 2, pp. 153–161, 2005.

[80] Grewe, L. and Kak, A.C.: Interactive learning of a multiple-attribute hash table classifier for fast object recognition. *Computer Vision and Image Understanding*, vol. 61, no. 3, pp. 387–416, 1995.

[81] Papageorgiou, C.P., Oren, M. and Poggio, T.: A general framework for object detection. In: *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*, pp. 555–562. IEEE, 1998.

[82] Girshick, R.: Fast R-CNN. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440–1448. 2015.

[83] Ren, S., He, K., Girshick, R. and Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: *Advances in Neural Information Processing Systems*, pp. 91–99. 2015.

[84] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y. and Berg, A.C.: SSD: Single shot multibox detector. In: *European Conference on Computer Vision*, pp. 21–37. Springer, 2016.

[85] Redmon, J., Divvala, S., Girshick, R. and Farhadi, A.: You Only Look Once: Unified, real-time object detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779–788. 2016.

[86] Goodfellow, I., Bengio, Y. and Courville, A.: *Deep learning*, vol. 1. MIT press Cambridge, 2016.

[87] Sachan, A.: Zero to hero: guide to object detection using deep learning: Faster R-CNN, YOLO, SSD. https://cv-tricks.com/object-detection/faster-r-cnn-yolo-ssd/. Accessed on: June 22, 2020.

[88] He, K., Zhang, X., Ren, S. and Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[89] Kalal, Z., Mikolajczyk, K. and Matas, J.: Forward-backward error: Automatic detection of tracking failures. In: *2010 20th International Conference on Pattern Recognition*, pp. 2756–2759. IEEE, 2010.

[90] Comaniciu, D., Ramesh, V. and Meer, P.: Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 564–577, 2003.

[91] Grabner, H., Grabner, M. and Bischof, H.: Real-time tracking via on-line boosting. In: *British Machine Vision Conference 2006*, vol. 1, pp. 47–56. 2006.

[92] Freund, Y. and Schapire, R.E.: A desicion-theoretic generalization of on-line learning and an application to boosting. In: *European Conference on Computational Learning Theory*, pp. 23–37. Springer, 1995.

[93] Ojala, T., Pietikainen, M. and Maenpaa, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[94] Babenko, B., Yang, M.-H. and Belongie, S.: Visual tracking with online multiple instance learning. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 983–990. IEEE, 2009.

[95] Kalal, Z., Mikolajczyk, K. and Matas, J.: Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1409–1422, 2011.

[96] Sato, K. and Aggarwal, J.K.: Temporal spatio-velocity transform and its application to tracking and interaction. *Computer Vision and Image Understanding*, vol. 96, no. 2, pp. 100–128, 2004.

[97] Li, B., Chellappa, R., Zheng, Q. and Der, S.Z.: Model-based temporal object verification using video. *IEEE Transactions on Image Processing*, vol. 10, no. 6, pp. 897–908, 2001.

[98] Yilmaz, A., Li, X. and Shah, M.: Contour-based object tracking with occlusion handling in video acquired using mobile cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 11, pp. 1531–1536, 2004.

[99] Cremers, D., Kohlberger, T. and Schnörr, C.: Nonlinear shape statistics in mumford-shah based segmentation. In: *European Conference on Computer Vision*, pp. 93–108. Springer, 2002.

[100] Chen, Z., Hong, Z. and Tao, D.: An experimental survey on correlation filter-based tracking. *arXiv preprint arXiv:1509.05520*, 2015.

[101] Bolme, D.S., Beveridge, J.R., Draper, B.A. and Lui, Y.M.: Visual object tracking using adaptive correlation filters. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2544–2550. IEEE, 2010.

[102] Henriques, J.F., Caseiro, R., Martins, P. and Batista, J.: High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2014.

[103] Lukežič, A., Vojíř, T., Čehovin Zajc, L., Matas, J. and Kristan, M.: Discriminative correlation filter with channel and spatial reliability. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6309–6318. 2017.

[104] Ning, G., Zhang, Z., Huang, C., Ren, X., Wang, H., Cai, C. and He, Z.: Spatially supervised recurrent convolutional neural networks for visual object tracking. In: *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–4. IEEE, 2017.

[105] wcUAVc: Wildlife conservation UAV challenge. https://www.wcuavc.com/. Accessed on: Feb 11, 2020.

[106] van Gemert, J.C., Verschoor, C.R., Mettes, P., Epema, K., Koh, L.P. and Wich, S.: Nature conservation drones for automatic localization and counting of animals. In: *European Conference on Computer Vision*, pp. 255–270. Springer, 2014.

[107] Fang, Y., Du, S., Abdoola, R., Djouani, K. and Richards, C.: Motion based animal detection in aerial videos. *Procedia Computer Science*, vol. 92, pp. 13–17, 2016.

[108] Łoza, A., Mihaylova, L., Bull, D. and Canagarajah, N.: Structural similarity-based object tracking in multimodality surveillance videos. *Machine Vision and Applications*, vol. 20, no. 2, pp. 71–83, 2009.

[109] Webb, A.R.: *Statistical pattern recognition*. John Wiley & Sons, 2003.

[110] Wang, F., Łoza, A., Yang, J., Xue, Y. and Yu, S.: Surveillance video object tracking with differential SSIM. In: *Proc. IEEE Intl. Conf. on Multimedia and Expo., Cancun, Mexico (June 2009)*.

[111] Sampat, M.P., Wang, Z., Gupta, S., Bovik, A.C. and Markey, M.K.: Complex wavelet structural similarity: A new image similarity index. *IEEE Transactions on Image Processing*, vol. 18, no. 11, pp. 2385–2401, 2009.

[112] Xu, K., Liu, X., Cai, H. and Gao, Z.: Full-reference image quality assessment-based b-mode ultrasound image similarity measure. *arXiv preprint arXiv:1701.02797*, 2017.

[113] Yavariabdi, A. and Kusetogullari, H.: Change detection in multispectral landsat images using multiobjective evolutionary algorithm. *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 3, pp. 414–418, 2017.

[114] Acuna, D.: Towards real-time detection and tracking of basketball players using deep neural networks. In: *31st Conference on Neural Information Processing Systems (NIPS 2017)*. 2017.

[115] Bewley, A., Ge, Z., Ott, L., Ramos, F. and Upcroft, B.: Simple online and realtime tracking. In: *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 3464–3468. IEEE, 2016.

[116] Kim, S., Nam, J. and Ko, B.: Multiple pedestrian tracking in moving vehicle using online learning of random ferns and feature descriptor of pre-trained shallow convolutional neural networks. *Electronic Imaging*, vol. 2018, no. 17, pp. 347–1, 2018.

[117] Kwak, J.-Y., Ko, B.C. and Nam, J.Y.: Pedestrian tracking using online boosted random ferns learning in far-infrared imagery for safe driving at night. *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 1, pp. 69–81, 2016.

[118] Kong, A., Liu, J.S. and Wong, W.H.: Sequential imputations and Bayesian missing data problems. *Journal of the American Statistical Association*, vol. 89, no. 425, pp. 278–288, 1994.

[119] Liu, J.S.: Metropolized independent sampling with comparisons to rejection sampling and importance sampling. *Statistics and Computing*, vol. 6, no. 2, pp. 113–119, 1996.

[120] Doucet, A., Godsill, S. and Andrieu, C.: On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, vol. 10, no. 3, pp. 197–208, 2000.

[121] Felzenszwalb, P.F., Girshick, R.B., McAllester, D. and Ramanan, D.: Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2009.

[122] Kathuria, A.: What's new in YOLOv3? https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b. Apr 23, 2018. Accessed on: July 19, 2020.

[123] Akkoyunlu, E.A., Ekanadham, K. and Huber, R.V.: Some constraints and tradeoffs in the design of network communications. In: *Proceedings of the Fifth ACM Symposium on Operating Systems Principles*, pp. 67–74. 1975.

[124] Gray, J.N.: Notes on data base operating systems. In: *Operating Systems*, pp. 393–481. Springer, 1978.

[125] Lamport, L., Shostak, R. and Pease, M.: The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382–401, 1982.

[126] Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Tech. Rep., Manubot, 2019.

[127] Fagin, R., Moses, Y., Halpern, J.Y. and Vardi, M.Y.: *Reasoning about knowledge*. MIT press, 2003.

[128] Patacchiola, M. and Cangelosi, A.: Head pose estimation in the wild using convolutional neural networks and adaptive gradient methods. *Pattern Recognition*, vol. 71, pp. 132–143, 2017.

[129] Grabner, H. and Bischof, H.: On-line boosting and vision. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 1, pp. 260–267. IEEE, 2006.

[130] Gustafsson, F., Gunnarsson, F., Bergman, N., Forssell, U., Jansson, J., Karlsson, R. and Nordlund, P.-J.: Particle filters for positioning, navigation, and tracking. *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 425–437, 2002.

[131] Wolpert, D.H.: The lack of a priori distinctions between learning algorithms. *Neural Computation*, vol. 8, no. 7, pp. 1341–1390, 1996.

[132] Wolpert, D.H. and Macready, W.G.: No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.